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

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

DISSERTAÇÃO DE MESTRADO

Fernando Stefanello

Santa Maria, RS, Brasil
2011

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

por

Fernando Stefanello

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Informática, Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientador: Felipe Martins Müller

Santa Maria, RS, Brasil

2011

S816h Stefanello, Fernando
Hibridização de métodos exatos e heurísticos para resolução de problemas de otimização combinatória / por Fernando Stefanello. – 2011.
88 f. ; il. ; 30 cm

Orientador: Felipe Martins Müller
Dissertação (mestrado) – Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2011

1. Informática 2. Computação 3. Programação 4. Heurística 5. Otimização combinatória 6. Processamento paralelo I. Müller, Felipe Martins II. Título.

CDU 004

Ficha catalográfica elaborada por Cláudia Terezinha Branco Gallotti – CRB 10/1109
Biblioteca Central UFSM

Endereço: Canabarro - Boca do Monte, Santa Maria, RS, CEP: 97170-000
Fone (0xx)55 9606-4361; End. Eletr: stefanello@inf.ufsm.br /fernandostefa@gmail.com

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

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

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

elaborada por
Fernando Stefanello

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Felipe Martins Müller, Dr.
(Presidente/Orientador)

Michel Gendreau, Ph.D.
(CIRRELT, Universidade de Montreal)

José Vicente Canto dos Santos, Dr.
(UNISINOS)

Santa Maria, 04 de março de 2011.

A minha família, e meus afilhados.

AGRADECIMENTOS

Escrever os agradecimentos de um trabalho é um momento muito esperado, no entanto, muito difícil. Foram tantas as lições, as alegrias e tristezas, os momentos de dúvida, certezas e incertezas, todas em algum momento compartilhadas com pessoas presentes em minha vida. Corro aqui um sério risco de deixar algum momento ou alguém que tanto merece passar sem o devido reconhecimento, no entanto, se por ventura isso acontecer, sintam-se profundamente agradecidos.

Primeiramente quero agradecer a Deus, pelo dom da vida, e por me proteger, dar força e me acalmar nos momentos difíceis e conturbados que passei, e lógico, pelos felizes que me proporcionou.

Ao meu orientador, Prof. Felipe Martins Müller, por ter confiado na minha capacidade possibilitado o ingresso no mestrado, além de suas orientações e ajudas sempre que necessário.

Ao Professor Olinto César Bassi de Araújo. Nem com folhas e folhas de escrita conseguirei expressar meu reconhecimento, minha gratidão e a importância de seu auxilio no decorrer dos estudos desta dissertação.

A toda minha família. Vocês foram a base forte nos momentos difíceis, e sem vocês isso não teria o menor sentido. Mais especificamente agradeço aos meus pais Edemar e Clede, quero dizer que se fisicamente não pude estar mais presente, meu coração sempre esteve e estará com vocês. Vocês me deram o aprendizado que jamais encontraria em anos de pesquisa nos livros.

Aos meus irmãos Edevandro e Sandra, o apoio de vocês foi mais que decisivo para que hoje eu pudesse estar escrevendo isso. Vocês são especiais!

Aos meus cunhados Maicon e Gisele. Vocês são mais que cunhados, são meus irmãos.

Aos meus afilhados Gustavo e Gabriela. Mesmo que hoje vocês não tenham a devida noção, vocês representam muito para mim.

A todos os meus amigos e amigas que estiveram presentes me aconselhando e incentivando com carinho e dedicação.

Aos que em algum momento fizeram parte de minha vida. Carrego de vocês as melhores frases e os melhores momentos, os quais ajudaram a construir meu caráter e minha formação.

A todas as pessoas que, direta ou indiretamente, contribuíram de uma forma ou de outra para a conclusão desta dissertação de mestrado, o meu mais profundo e sincero MUITO OBRIGADO.

Os números governam o mundo

(Platão)

RESUMO

Dissertação de Mestrado

Programa de Pós-Graduação em Informática

Universidade Federal de Santa Maria

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

AUTOR: FERNANDO STEFANELLO

ORIENTADOR: FELIPE MARTINS MÜLLER

Data e Local da Defesa: Santa Maria, 04 de março de 2011.

A recente evolução dos computadores como também dos métodos exatos oriundos da programação matemática, muitos destes eficientemente implementados em otimizadores comerciais, propiciou o surgimento de novos algoritmos, denominados metaheurísticas híbridas, que têm sido aplicados para resolução de problemas combinatoriais. Este trabalho apresenta abordagens que hibridizam metaheurísticas baseadas em busca local com algoritmos exatos de programação matemática para resolver dois problemas de otimização combinatória. Mais especificamente, para o primeiro problema, o problema das p-medianas capacitado, a proposta considera a eliminação heurística de variáveis do modelo matemático, que permite a obtenção de soluções de boa qualidade em um curto tempo computacional, e a combinação com um procedimento iterativo no qual apenas um determinado subconjunto de pontos é considerado. No que se refere ao segundo problema, programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina com objetivo de minimizar o tempo de processamento total da máquina com maior carga entre todas (*makespan*), propõe-se um modelo matemático para varrer a vizinhança de uma solução e identificar sequências de movimentos de tarefas que podem ser aplicadas na respectiva solução de modo a minimizar a função objetivo. Nos dois casos os modelos matemáticos são resolvidos utilizando um otimizador comercial. Extensivos testes computacionais são realizados para demonstrar o bom desempenho das abordagens propostas.

Palavras-chave: metaheurísticas híbridas, problema das p-medianas capacitado, problema de programação máquinas paralelas não relacionadas, vizinhança de grande porte.

ABSTRACT

Master's Dissertation

Post-Graduate Program in Informatics

Federal University of Santa Maria

HYBRIDIZATION OF EXACT AND HEURISTIC METHODS TO SOLVE COMBINATORIAL OPTIMIZATION PROBLEM

AUTOR: FERNANDO STEFANELLO

ADVISOR: FELIPE MARTINS MÜLLER

Place and Date of Presentation: Santa Maria, march 04, 2011.

The evolution of computer hardware as well as new applications of mathematical programming techniques, efficiently implemented in many commercial solvers, has given rise to new algorithms called hybrid metaheuristic, which have been applied to solve combinatorial problems. This work presents several approaches which try to deal with the hybridization of local search based metaheuristics with exact algorithms to solve two problems of combinatorial optimization. More specifically, the first problem, capacitated p-median problem, the proposed approach considers heuristic elimination of variable of the original mathematical model, that produce solutions of very good quality in a short amount of time, and a combination with an iterative procedure in which only a certain subset of points is considered. As regards the second problem, unrelated parallel machine scheduling with sequence and machine dependent setup time problem of minimizing makespan, is proposed a mathematical model to search the neighborhood of a solution and identify movement sequences to minimize the objective function. In both cases, mathematical models are solved using a commercial solver. Extensive computational experiments are carried out to demonstrate the good performance of the proposed approaches.

Keywords: hybrid metaheuristics, capacitated p-median problem, unrelated parallel machine scheduling problem, large-scale neighborhood.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Problemas de otimização combinatória estão presentes em muitas aplicações cotidianas, em sistemas produtivos em geral, como programação de tarefas em máquinas paralelas, localização de facilidades, construção de escalas de serviço, entre outros. Nos problemas de otimização combinatória o domínio é tipicamente finito e, em geral, é fácil listar os seus elementos e também testar se um dado elemento pertence a esse domínio. No entanto, a ideia ingênua de testar todos os elementos deste domínio na busca pelo melhor, mostra-se inviável na prática em determinadas classes de problemas, mesmo para instâncias de tamanho moderado, devido ao número exorbitante de soluções possíveis para serem avaliadas. Uma destas classes de problemas é denominada na teoria da complexidade como NP-*difícil*, o que significa que não são conhecidos algoritmos eficientes de resolução e, embora existam algoritmos exatos para resolver na otimalidade estes problemas, os mesmos são restritos a casos de pequeno porte devido ao tempo computacional elevado requerido para resolução.

Dado que a maioria dos casos reais, modelados como problemas de otimização combinatória envolvem um número grande de variáveis e o tempo para tomada de decisão em geral é pequeno, a inerente complexidade de resolução leva a situação em que é necessário sacrificar a obtenção da solução ótima em favor da obtenção de soluções de boa qualidade em um tempo razoável. Deste modo, é comum lançar mão de métodos heurísticos com a finalidade de encontrar boas aproximações da solução ótima em um curto tempo de processamento. Também é importante ressaltar que normalmente existe muita incerteza na coleta de dados e não raro a solução ótima da representação do problema não está ligada diretamente à solução ótima da situação real, com isto, uma solução aproximada do ótimo teórico é plenamente aceitável.

Com a evolução dos computadores nas últimas décadas, como também de otimizadores genéricos para problemas combinatoriais, observa-se uma crescente tendência da utilização de métodos híbridos na busca por soluções de problemas de otimização como relatado em Jourdan et al. (2009). Métodos híbridos consistem em utilizar métodos exatos e heurísticos combinados, de forma a potencializar as melhores características de cada um dos métodos no intuito de obter soluções de melhor qualidade para os problemas tratados.

Neste trabalho são abordadas duas estratégias distintas de hibridização de métodos exatos com métodos heurísticos para resolver problemas de otimização combinatória. A primeira, aplicada ao problema das p-medianas capacitado (PPMC) e a segunda, ao problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina com objetivo de minimizar o tempo de processamento total da máquina com maior carga entre todas (*makespan*).

No problema das p-medianas capacitado tem por objetivo criar agrupamentos associando pontos a medianas de modo que a soma das distâncias dos pontos as medianas seja minimizado e a soma das demandas dos pontos associados a uma mediana não exceda a capacidade da mesma. Este problema possui diversas aplicações práticas como localização de fábricas, hospitais, escolas e centros de coleta de lixo. Para a sua resolução, a estratégia de hibridização resume-se na eliminação heurística de variáveis do modelo matemático com menor probabilidade de pertencerem à solução ótima ou uma boa solução (processo denominado Redução de Variáveis), e posterior resolução do modelo matemático por um otimizador comercial sem as variáveis previamente eliminadas. Ainda para este problema, propõe-se um método que explora a vizinhança de uma solução a partir de um modelo matemático restrito.

O problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina com objetivo de minimizar o *makespan,* o objetivo é alocar tarefas em determinadas máquinas de modo a minimizar o tempo de processamento da máquina mais carregada. Este problema possui aplicações práticas em fábricas ou empresas onde o processamento de determinada tarefa exige um tempo de preparação da máquina. Tal preparação pode ser compreendida como um tempo para limpeza ou ajustes da máquina que irá processar a tarefa. Neste caso, as máquinas são diferentes entre si, implicando no fato de que o tempo de preparação de uma máquina para processamento de uma tarefa, também é diferenciado. Para a resolução deste problema é considerado um modelo matemático para explorar a vizinhança de uma solução e identificar sequências de movimentos que devem ser executados de modo a melhorar o valor da função objetivo do problema.

Para os dois problemas citados os respectivos modelos matemáticos são resolvidos a partir de um otimizador comercial como um procedimento do tipo “caixa preta”. No entanto, nada impede que algoritmos exatos sejam desenvolvidos considerando características intrínsecas de cada modelo.

1.1 Objetivo e estrutura do trabalho

O objetivo deste trabalho é propor e testar abordagens híbridas nas quais uma heurística engloba métodos exatos para a resolução de problemas de otimização combinatória.

O trabalho está estruturado em cinco capítulos. No Capítulo 2 é feita uma revisão sobre os métodos de resolução de problemas de otimização combinatória com ênfase nos métodos híbridos, expondo algumas taxonomias utilizadas para classificar estes métodos.

No Capítulo 3 é abordado o problema da p-medianas capacitado, onde é apresentado um procedimento para obtenção de um conjunto diverso de soluções, quatro estratégias de redução de variáveis e um algoritmo iterativo utilizado para exploração da vizinhança em determinadas regiões da solução. Ao final do capítulo, são apresentados diversos testes computacionais utilizando as diferentes estratégias de resolução propostas para cinco conjuntos de instâncias.

O Capítulo 4 aborda o problema de programação de tarefas em máquinas paralelas, mais especificamente de máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina, com objetivo de minimizar o *makespan*. Inicialmente o problema é apresentado juntamente com sua formulação matemática e duas heurísticas construtivas. Em seguida, é apresentada uma vizinhança de grande porte baseada em programação inteira mista com um método de redução de variáveis e uma estratégia de diversificação. Esta vizinhança é utilizada internamente numa busca local e num algoritmo de múltiplos inícios descritos no mesmo capítulo. Por fim, são reportados extensivos testes computacionais que comprovam a eficiência do método proposto.

No Capítulo 5 são apresentadas as conclusões e trabalhos futuros e por último, são apresentadas as referências bibliográficas.

2 HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS

A otimização combinatória tem como objetivo a resolução de problemas referentes à alocação de recursos, tipicamente limitados, com o intuito de alcançar determinados objetivos. Problemas deste tipo surgem em várias áreas práticas, tais como desenvolvimento de circuitos digitais, programação de tarefas, localização de facilidades, reconhecimento de padrões, planejamento de transporte, entre outros. A importância desses problemas para indústria não pode ser subestimada. Por exemplo, Anbil et al. (1991) descreve um problema de escalas de tripulações em uma empresa aérea cujos custos de operação são da ordem de bilhões de dólares, e mesmo pequenas melhoras na eficiência da alocação de pessoal podem levar a substanciais economias.

Problemas de otimização combinatória (POC) são normalmente fáceis de descrever, mas difíceis de resolver. A dificuldade reside no objetivo de encontrar uma solução que receba a melhor avaliação possível e, ao mesmo tempo, consiga satisfazer todas as restrições impostas. Esta solução é denominada solução ótima. Considerando que existe um conjunto discreto e, usualmente, finito de soluções, um problema de otimização combinatória pode ser resolvido a partir do processo de geração, avaliação e comparação de soluções. Deste modo, ao examinar exaustivamente todos os elementos do domínio do problema é possível sempre encontrar a melhor solução ou concluir que esta não existe. Portanto, qualquer problema pode ser resolvido em princípio, mas isto não significa que possa ser resolvido em tempo aceitável. Este tema em específico é abordado na Teoria da Complexidade, que trata da eficiência da computação de algoritmos em computadores existentes e os classifica segundo a dificuldade de resolução. Em Arora e Barak (2009) é apresentado um estudo mais aprofundado no assunto.

Dado que muitos POC são caracterizados como *NP-difíceis* (Garey & Johnson, 1979), a otimização combinatória tem sido um campo desafiador para pesquisadores vindos de diferentes áreas, incluindo matemática, ciência da computação, engenharia, economia e administração.

Existem numerosas abordagens para resolver POC, e duas correntes, oriundas de diferentes comunidades científicas, têm alcançado significativo sucesso:

- Programação inteira como uma abordagem exata, oriunda da comunidade de pesquisa operacional e baseada em conceitos de programação linear.

- Busca local com várias extensões e variantes desenvolvidas independentemente, denominadas metaheurísticas, como uma abordagem heurística.

Mais recentemente, em grande parte devido a evolução dos computadores, vários estudos vêm sendo desenvolvidos no sentido de hibridizar métodos exatos e heurísticos no intuito de explorar as vantagens de cada um dos métodos.

Neste capítulo são referenciados alguns métodos exatos, heurísticos e híbridos para resolução de POC, com ênfase no último, pois é o principal método abordado no decorrer deste trabalho.

2.1 Métodos exatos

Uma característica importante nestes métodos é a garantia da obtenção da solução ótima do problema, no entanto estes métodos costumam ser eficientes apenas em instâncias de pequeno e médio porte, pois o tempo de execução, frequentemente, aumenta dramaticamente com o tamanho da instância, o que restringe o uso prático destes algoritmos.

Entre os métodos exatos estão programação dinâmica, métodos baseados em relaxação lagrangeana, e métodos baseados em programação linear e inteira, tais como *branch-and-bound*, *branch-and-cut*, *branch-and-price* e *branch-and-cut-and-price* (Nemhauser e Wolsey 1988). Muitas destas técnicas são projetadas para serem flexíveis e independentes de domínio, a fim de serem aplicáveis a uma grande variedade de problemas práticos sem o uso intensivo de estratégias específicas. Pesquisas sobre técnicas independentes do domínio para otimização combinatória têm resultado em ferramentas de propósito geral para programação inteira mista, tal como a variedade de otimizadores baseados em *branch-and-bound* (CPLEX[1], LINDO[2], XPRESS[3], GLPK[4], para citar alguns). Uma vez que estas técnicas trabalham com modelos matemáticos como representação de um problema, proporcionam a necessária flexibilidade para responder prontamente à troca de requisitos que frequentemente ocorre em ambientes reais. Além disso, normalmente usuários carecem de tempo e conhecimento para pesquisarem e desenvolverem algoritmos específicos.

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
[2] http://www.lindo.com/
[3] http://www.aimms.com/features/solvers/xpress
[4] http://www.gnu.org/software/glpk/

2.2 Métodos heurísticos

Heurísticas construtivas são métodos de obtenção de uma solução inicial para um determinado problema, que constroem uma solução elemento a elemento seguindo uma sequência definida de passos. Quando uma heurística construtiva avalia a inserção do próximo elemento tomando a decisão de inserir o "melhor", ou seja, o que aumenta menos o custo da solução, esta heurística construtiva é chamada de gulosa ou míope. Escolhas desse tipo podem levar a escolhas de maior custo em passos seguintes, gerando soluções de baixa qualidade, mas em tempos computacionais aceitáveis.

Metodologias baseadas em metaheurísticas englobam técnicas como *simulated annealing*, busca tabu, *iterated local search*, busca em vizinhança variável, algoritmos genéticos e *scatter search*, para citar alguns. Em Blum e Roli (2003) os autores apresentam uma visão geral do assunto. Como regra, estas técnicas são especializadas para resolver uma classe específica de problemas de otimização, diferentemente dos métodos exatos que são mais abrangentes. Nas metaheurísticas, a qualidade da solução é um fator crucial e o tempo de projeto e desenvolvimento do algoritmo é, em geral, maior.

A aplicação de uma metaheurística é, em muitos casos, a única possibilidade para modelar e encontrar boas soluções para instâncias grandes. Neste caso, a garantia de encontrar soluções ótimas é sacrificada em prol de se obter boas soluções em um tempo razoável. Para desenvolver algoritmos específicos satisfatórios para um dado problema é necessário explorar estruturas matemáticas especiais, o que demanda grande quantidade de tempo e energia. Além disso, frequentemente é o caso de um algoritmo especificamente projetado para um problema em particular não ser útil se novas restrições são adicionadas ou a função objetivo é ligeiramente modificada.

2.3 Métodos híbridos

A evolução dos computadores nos últimos anos, caracterizada pelo aumento sistemático da capacidade de processamento e avanço nas técnicas de paralelização, tem permitido a avaliação e solução de problemas computacionais num tempo inferior se comparado com os computadores de poucos anos atrás. Soma-se a esse cenário a melhora no desempenho dos otimizadores comerciais para programação inteira, cuja eficiência não pode

ser desconsiderada aos olhos da pesquisa. Ainda, segundo Nievergelt (2000), é aceito que não existe algoritmo capaz de resolver inteiramente uma classe de problemas NP-*difíceis*, mas ignora-se o fato de que muitas instâncias dos problemas desta classe, inclusive as de interesse prático, podem ser resolvidas eficientemente.

Este cenário torna propício a hibridização de métodos exatos e heurísticos de forma a explorar as melhores características de cada um destes dois métodos já que a maioria das metaheurísticas são rápidas e especializadas enquanto os métodos exatos são demorados e, em muitos casos, genéricos. Em Jourdan et al. (2009), os autores apresentam um levantamento e classificação de várias abordagens encontradas na literatura que fazem uso de hibridização, mostrando o número crescente de publicações relacionadas (Figura 1), o que comprova o interesse que este assunto vem despertando na comunidade científica.

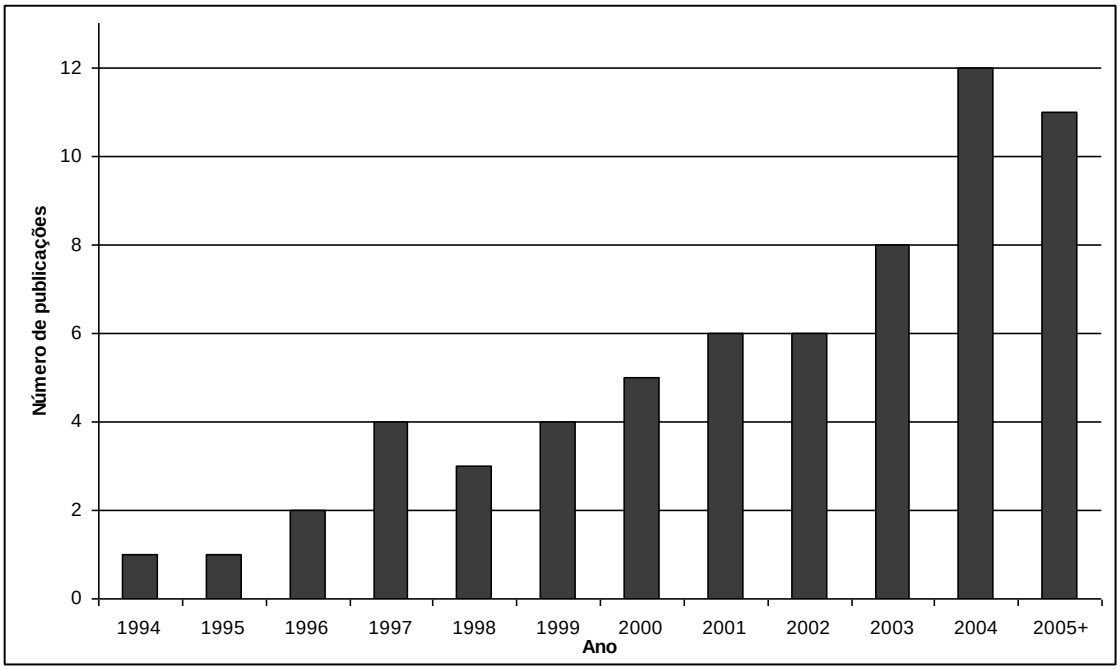

Figura 1 – Evolução da quantidade de publicações que propõem abordagens que hibridizam métodos exatos e heurísticos.

Além disso, os autores apresentam uma extensão do trabalho realizado em Talbi (2002), no qual, além de fazer uma breve revisão da literatura, também mostram como os métodos podem ser combinados e como pode ser a cooperação entre si. Segundo a classificação, a taxonomia é dividida em três aspectos gerais: projeto de métodos cooperativos

(*cooperation method design*), projeto de modelos de abordagens (*approach design*) e questões de implementação (*implementation issues*). Uma revisão considerando vários trabalhos é apresentada de acordo com a taxonomia proposta.

Em Dumitrescu e Stützle (2003), os autores distinguem cinco classes para a cooperação entre os métodos exatos e busca local, que são:

- Utilizar algoritmo exato para exploração da vizinhança de grande porte num algoritmo de busca local.
- Realizar várias execuções de uma busca local e explorar as informações em soluções de alta qualidade para definir problemas menores que são passíveis de solução com algoritmos exatos.
- Explorar limitantes em heurísticas construtivas.
- Utilizar as informações da relaxação de problemas de programação inteira para guiar a busca local ou algoritmos construtivos.
- Utilizar algoritmos exatos para procedimentos específicos dentro de metaheurísticas híbridas.

No entanto, esta classificação não aborda todos os métodos de otimização e até mesmo exclui algumas combinações, como o pré-processamento.

Em Puchinger e Raidl (2005), os autores classificam as combinações de métodos exatos e metaheurísticas em duas classes. Na primeira classe algoritmos distintos trocam informações entre si, ou seja, um não contém o outro, e são denominados algoritmos colaborativos. Esta classe, por sua vez, é subdividida em execução sequencial e execução paralela e entrelaçada. Na execução sequencial uma das técnicas realiza o processamento inicial servindo como base para o subsequente processamento pela outra técnica. Na segunda subclasse, os algoritmos são executados simultaneamente e atuam como equipes trocando informações entre si.

A segunda classe envolve os algoritmos de combinação integrada, ou seja, os algoritmos que possuem uma técnica como um componente subordinado ou encaixado dentro da outra. Esta classe subdivide-se também em duas subclasses, sendo a primeira a incorporação de métodos exatos em metaheurísticas e a segunda a incorporação de metaheurísticas em métodos exatos.

A primeira subclasse dos algoritmos de combinação integrada é subdividida em quatro novos grupos descritos a seguir:

- Resolução de problema relaxado – solução para relaxação heurística guiando busca na vizinhança, recombinação, mutação, reparação e/ou busca local
- Busca em vizinhanças de grande porte – algoritmo exato é usado na busca da vizinhança na busca local baseados em metaheurísticas.
- Fusão de soluções – o algoritmo exato é usado para resolver problemas secundários que geram soluções parciais. Estas soluções parciais são fundidas e aplicadas iterativamente dentro de uma metaheurística.
- Decodificação – em métodos evolutivos, um algoritmo exato é utilizado para encontrar a melhor solução que corresponde a um cromossomo que representada soluções de forma incompleta.

A segunda subclasse, em que metaheurísticas são incorporadas em métodos exatos, também é subdivida em quatro grupos:

- Atualização de limitantes – metaheurísticas são utilizadas para atualizar, durante o processo, as soluções incumbentes ou limitantes de métodos exatos.
- Geração de colunas e cortes – em algoritmos como *branch-and-price* e *branch-and-cut*, metaheurísticas são usadas para definir dinamicamente as colunas e planos de corte, respectivamente.
- Orientação estratégica de algoritmos exatos – metaheurísticas são utilizadas para determinar cortes estratégicos em técnicas *branch-and-bound*.
- Aplicação do espírito de metaheurística – o próprio *branch-and-bound* é usado para fazer a busca local. Explicitamente, nenhuma metaheurística é usada.

Uma representação gráfica da divisão de classes e subclasses da classificação de Puchinger e Raidl (2005) pode ser observada na Figura 2.

Em Fernandes e Lourenço (2007), as autoras apresentam um mapeamento dos trabalhos que usam procedimentos híbridos baseados na combinação de metaheurísticas e métodos exatos levando em consideração as classificações propostas por Dumitrescu e Stützle (2003) e Puchinger e Raidl (2005).

Nos trabalhos Mautor e Michelon (1997) e Mautor e Michelon (2001) os autores apresentam o método *MIMAUSA*. Este método foi usado no problema quadrático de alocação em que a busca local pela vizinhança é definida eliminando algumas variáveis, e resolvendo o subproblema de forma exata.

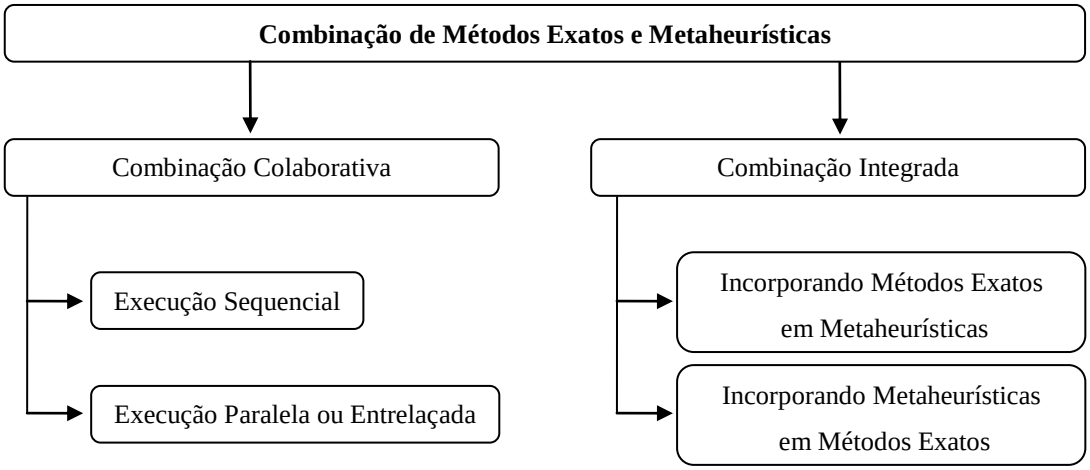

Figura 2 – Classes e subclasses da combinação entre métodos exatos e metaheurísticas segundo Puchinger e Raidl.

O trabalho de Fanjul-Peyro e Ruiz (2010) explora o fato que otimizadores comerciais são eficientes na resolução de instâncias de pequeno e médio porte e propõe um conjunto de heurísticas baseado em redução de variáveis do modelo matemático (em inglês, *size-reduction heuristic*). A ideia subjacente consiste em reduzir o tamanho do problema e, consequentemente a sua complexidade, para então utilizar um otimizador comercial. O problema tratado pelos autores é a programação de tarefas em máquinas paralelas não relacionadas e, para este problema, a estratégia consiste em considerar somente algumas máquinas, não todas, para programação de uma determinada tarefa. Esta regra reduz drasticamente o número de variáveis binárias, proporcionando obtenção de soluções sub-ótimas em tempo computacional reduzido. Obviamente, dada a natureza heurística, não é possível garantir a otimalidade, mas possibilita a obtenção de soluções de boa qualidade com a utilização de um otimizador comercial como "caixa preta". Técnica semelhante já havia sido aplicada na resolução do PPMC em Stefanello e Müller (2009) dando origem a este trabalho.

Do exposto, é possível dizer que as metodologias propostas neste trabalho classificam-se como a utilização de algoritmos exatos para procedimentos específicos, internos ou sequenciais à procedimentos heurísticos. De forma mais abrangente, como utilização de heurísticas que utilizam métodos exatos para resolução de subproblemas, denominado *Optimised Search Heuristics* (OSH) por Fernandes e Lourenço (2007).

3 PROBLEMA DAS P-MEDIANAS CAPACITADO

3.1 Introdução

Problemas de agrupamento capacitado possuem diversas aplicações práticas, como a localização de fábricas, hospitais, escolas e centros de coleta de lixo, entre outros. Em todos esses casos, dado um conjunto de nós associado à realidade que se deseja modelar, o objetivo é agrupá-los de forma a maximizar a dissimilaridade entre os agrupamentos. Nesse contexto, ressalta-se o Problema das p-Medianas Capacitado (PPMC). O PPMC trata da alocação de *p* facilidades (medianas) para servir *n* pontos de demandas (nós). O objetivo é minimizar a soma das distâncias entre os nós e as medianas, com a restrição de não exceder a capacidade de cada mediana instalada.

Os primeiros trabalhos que tratam do PPMC apareceram na literatura científica na década de 80, com os estudos de Mulvey e Beck (1984). Em Osman e Christofides (1994) os autores utilizam um método híbrido combinando *simulated annealing* e busca tabu. Em Maniezzo et al. (1998) é utilizado um método evolutivo para resolução do PPMC. Mais recentemente, Baldacci et al. (2002) utilizam um novo método baseado na formulação de particionamento de conjuntos. Lorena e Senne (2002) apresentam um método de geração de colunas. Ahmadi e Osman (2005) propõem uma composição de metaheurísticas em um *framework* denominado GRAMPS (*Greedy Random Adaptive Memory Search Method*). Uma abordagem *scatter search* é utilizada por Scheuerer e Wendolsky (2006) e Diaz e Fernandez (2006). Fleszar e Hindi (2008) resolvem o PPMC em duas etapas, a primeira utiliza a metaheurística VNS (*Variable Neighborhood Search*) introduzida por Mladenovic e Hansen (1997) para definir conjuntos de medianas, e a segunda o software CPLEX para o problema de atribuição. Em Chaves et al. (2007) os autores apresentam uma heurística híbrida chamada *Clustering Search* (CS), que consiste em detectar áreas promissoras de busca baseada em agrupamentos. Boccia et al. (2008) propõem um algoritmo de planos de corte que utiliza o pacote de otimização CPLEX.

Esse problema é reconhecidamente NP-*difícil* (Garey & Johnson, 1979), o que sugere a inviabilidade do uso de métodos exatos para a sua resolução. Diante disso, vários estudos

utilizam métodos heurísticos para encontrar soluções de boa qualidade em tempo computacional aceitável para uma tomada de decisão. No entanto, é importante observar que a intratabilidade da questão não necessariamente ocorre em todas as instâncias do problema, e uma escolha criteriosa do conjunto de instâncias deve ser feita para demonstrar o desempenho dos métodos heurísticos.

Este capítulo apresenta os estudos realizados com modelos matemáticos e o uso de otimizadores comerciais para tratar o problema das p-medianas capacitado. Inicialmente, o problema é apresentando e definido formalmente. Na sequência, é proposto um método de resolução do PPMC que envolve um gerador diverso de soluções, quatro estratégias de redução de variáveis do modelo matemático e um algoritmo iterativo para lidar com as instâncias de grande porte, de modo a possibilitar a obtenção de soluções de boa qualidade em tempos aceitáveis. A seguir, vários experimentos computacionais são detalhados para os diferentes conjuntos de instâncias.

3.2 Modelo matemático

O PPMC é um problema de alocação de facilidades no qual, dado um grafo $G = (V, E)$, em que V representa o conjunto de nós ou pontos de interesse e E o conjunto de arcos com as distâncias (ou custos). A cada nó i, $i \in E$, é associado uma demanda q_i que representa o quanto o nó consome de recurso de sua respectiva mediana. Também é associado a um candidato a mediana j, $j \in E$, uma capacidade Q_j que aplica um limite de demanda que este candidato a mediana pode suportar. O objetivo é encontrar p subconjuntos de vértices $Vp \subseteq V$ tal que a soma das distâncias de cada vértice restante até a sua correspondente mediana em Vp seja a menor possível e a restrição de capacidade seja satisfeita. As primeiras formulações do problema foram apresentadas em Hakimi (1964).

O PPMC pode ser formulado matematicamente conforme segue:

Parâmetros

n: número de pontos;

p: número de medianas ou facilidades;

$I=\{1,2,...,n\}$: conjunto de nós ou pontos de demanda;

$J=\{1,2,...,n\}$: conjunto de candidatos a receber a instalação de uma mediana;

d_{ij}: distância entre os nós i e j;

q_i: demanda do ponto i;

Q_j: capacidade da mediana j.

Variáveis

$$w_{ij} = \begin{cases} 1 & \text{se o nó } i \text{ é designado ao agrupamento } j \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático

$$Min \sum_{i \in I} \sum_{j \in J} d_{ij} w_{ij} \tag{1}$$

Sujeito a:

$$\sum_{j \in J} w_{jj} = p \tag{2}$$

$$\sum_{j \in J} w_{ij} = 1 \qquad \forall i \in I \tag{3}$$

$$w_{ij} \leq w_{jj} \qquad \forall i \in I,\ \forall j \in J \tag{4}$$

$$\sum_{i \in I} q_i w_{ij} \leq Q_j w_{jj} \qquad \forall j \in J \tag{5}$$

$$w_{ij} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J \tag{6}$$

A função objetivo, que determina que a soma das distâncias dos nós até a respectiva mediana deve ser minimizada, é definida em (1). A restrição (2) assegura a existência de exatamente p agrupamentos e as restrições (3) e (4) determinam que cada nó é designado a apenas uma mediana. A restrição (5) limita a capacidade das medianas e a restrição (6) corresponde à condição de binariedade das variáveis.

As restrições (4) e (5) são redundantes, mas é conhecido que a restrição (4) melhora o desempenho de métodos baseados na relaxação da restrição de binariedade. O modelo matemático que faz uso da restrição (4) é referenciado no texto como modelo forte, enquanto o modelo que não usa tal restrição é denominado modelo fraco.

3.3 Métodos de resolução para o PPMC

Na literatura, vários métodos de resolução são propostos para o PPMC, tanto com viés heurístico quanto exato. Embora os métodos heurísticos proporcionem a possibilidade de resolver instância de grande porte, não há garantia que a solução encontrada seja ótima. Já os métodos exatos possuem esta propriedade, mas a aplicação destes restringe-se apenas a instâncias de pequeno e médio porte devido a complexidade inerente aos POC.

A utilização de otimizadores comerciais de propósito geral vem ganhando destaque diante de sua evolução concomitante com a evolução dos computadores nas últimas décadas. Sua utilização tem proporcionado a resolução na otimalidade de instâncias de pequeno e médio porte, exceto em certos casos onde mesmo para instâncias de pequeno porte a resolução pode ser inviável dependendo de certas características da instância, como apresentado nos trabalhos de Stefanello et al. (2009a) e Stefanello et al. (2009b). No entanto, dificilmente estas instâncias representam alguma situação prática considerando a aplicabilidade do problema.

Ainda que atualmente a resolução de POC através de otimizadores comerciais seja uma opção cada vez mais viável, para instâncias de grande porte torna-se inviável devido à grande quantidade de variáveis envolvidas. Como exemplo, para o PPMC, um arquivo em modo texto contendo as restrições do modelo matemático para instâncias com 3038 nós pode chegar a 800 MB de tamanho em disco. No entanto, avaliando algumas características do problema, é possível observar que certas variáveis do modelo matemático possuem uma probabilidade mínima de pertencerem a uma boa solução. O uso da eliminação destas variáveis do modelo matemático por meio de um pré-processamento possibilita que sejam consideradas apenas as principais variáveis possibilitando que o otimizador consiga resolver instâncias de maior porte.

A realização de um pré-processamento no qual determinadas variáveis são heuristicamente excluídas do modelo matemático é referida neste trabalho por Redução de Variáveis, e o modelo matemático resultante desta estratégia é denominado Modelo Reduzido, enquanto o modelo matemático contendo todas as variáveis é denominado Modelo Completo.

Algumas estratégias para eliminar heuristicamente variáveis com poucas chances de estarem envolvidas na solução ótima, ou ao menos possibilitar que sejam encontradas boas soluções utilizando unicamente o otimizador, além de uma heurística de obtenção de soluções iniciais e um algoritmo iterativo para busca por soluções são descritos a seguir.

3.3.1 Gerador diverso de soluções

Algumas estruturas de redução de variáveis propostas neste trabalho fazem uso da análise de um conjunto de soluções. Para a obtenção de soluções diversas é implementado um método de obtenção de solução baseado na heurística construtiva de Osman e Christofides (1994), método este que também serve para fornecer soluções iniciais.

O algoritmo para obtenção da solução é descrito nos passos a seguir:

Passo 1: Escolher aleatoriamente *p* pontos para serem medianas;

Passo 2: Selecionar aleatoriamente um nó dentre os que ainda não foram associados e alocar à mediana mais próxima que tenha capacidade disponível para atender sua demanda;

Passo 3: Caso não se obtenha uma solução factível, prossiga ao passo 6;

Passo 4: Recalcular o centro dos novos agrupamentos de forma a minimizar a função objetivo no agrupamento, ou seja, o nó que tiver o menor somatório das distâncias do nó a todos os outros nós do agrupamento será a nova mediana.

Passo 5: Atualizar a solução incumbente caso a atual seja melhor;

Passo 6: Caso o critério de parada seja atingido, retorne a solução incumbente;

Passo 7: Liberar todos os pontos não medianas de suas respectivas medianas e retornar ao Passo 2;

O critério de parada pode tanto ser o número de interações quanto o tempo computacional, ou ainda, um número de iterações ou recálculos de agrupamentos sem melhora da função objetivo.

Este algoritmo fornece soluções de boa qualidade em um tempo computacional reduzido, além de proporcionar uma desejável diversidade de soluções para alguns métodos de redução de variáveis descritos a seguir.

3.3.2 Redução baseada em demanda (R1)

Nesta estratégia de redução de variáveis, para todo nó *j* candidato a mediana, leva-se em conta apenas os *k* pontos mais próximos de *j*, cujo somatório das demandas seja inferior a uma dada capacidade atribuída ao nó, já subtraída de sua própria demanda, conforme segue.

$$\sum_{i \in K} q_i \leq \alpha Q_j - q_j \,, \qquad \text{(i)}$$

em que *K* é o conjunto de nós mais próximos do nó *j* que satisfazem a condição acima, com $|K| = k$, e α é um fator de expansão da capacidade usada para definir a quantidade de variáveis que são consideradas no modelo. Caso haja um ponto *i* tal que $q_i > Q_j - q_j$, então a variável que associa o ponto *i* à mediana *j* é automaticamente excluída.

A aplicação desta metodologia proporciona uma significativa redução do número de variáveis e, ainda que não garanta a otimalidade, possibilita o tratamento de problemas de maior porte.

A Figura 3 exemplifica a redução proposta, evidenciando quais pontos são passíveis de serem associados a dois pontos candidatos a medianas da instância sjc1 proposta por Lorena e Senne (2004), especificamente observado na Figura 3a. Também é apresentada a intersecção da solução ótima com a área de abrangência da redução (Figura 3b).

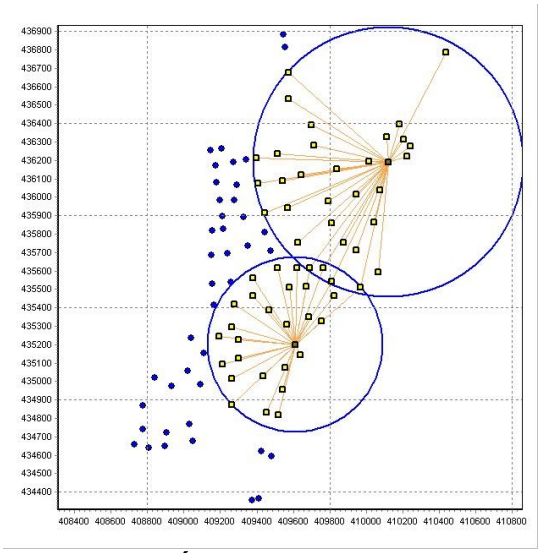

Figura 3a – Área e pontos que podem ser associados ao candidato a mediana.

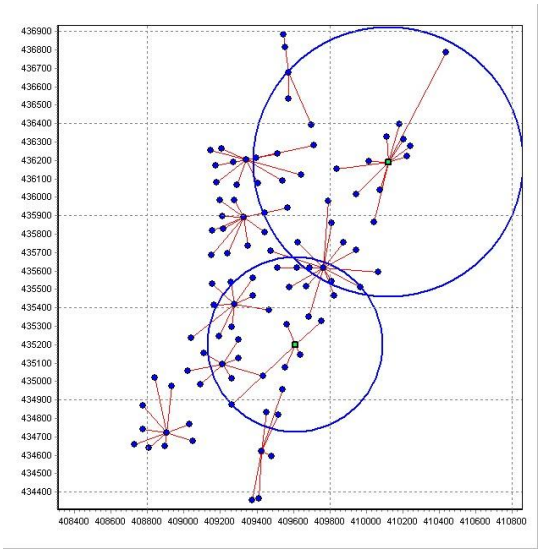

Figura 3b – Solução ótima, englobando todas as variáveis não extintas pela redução.

Figura 3 – Área de abrangência da redução para um candidato a mediana.

Este método de redução é mais eficiente quanto menor for a relação *n*/*p* onde *n* é o número de nós e *p* é o número de medianas da instância. Neste caso, menos pontos são candidatos a serem associados a cada possível mediana.

No caso descrito anteriormente, são excluídos os pontos para um determinado candidato a mediana, esta redução é denominada R1a. Existe a possibilidade de fazer a mesma análise de forma inversa. Tomando como referência um determinado ponto *i*, considera-se como candidatos a medianas para *i* apenas aqueles pontos que satisfazem a equação (i), e neste caso a redução é denominada R1b.

Apesar de serem muito semelhantes na concepção e manterem o mesmo número de variáveis eliminadas, é possível observar na Figura 4 que para os dois pontos candidatos a medianas do exemplo anterior, as diferenças são significativas. Nesta figura, as linhas indicam os pontos que podem ser associados aos candidatos a medianas. A Figura 4a mostra o pontos que podem ser associados às medianas segundo a redução R1a, enquanto a Figura 2b mostra os pontos que podem ser associados às medianas segundo a redução R1b.

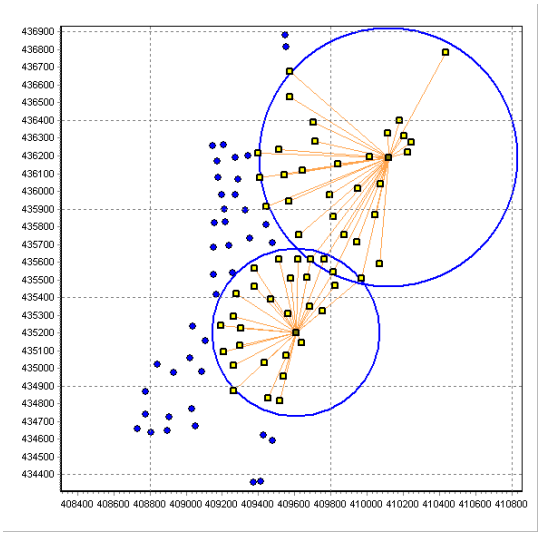

Figura 4a – Representação da redução R1a.

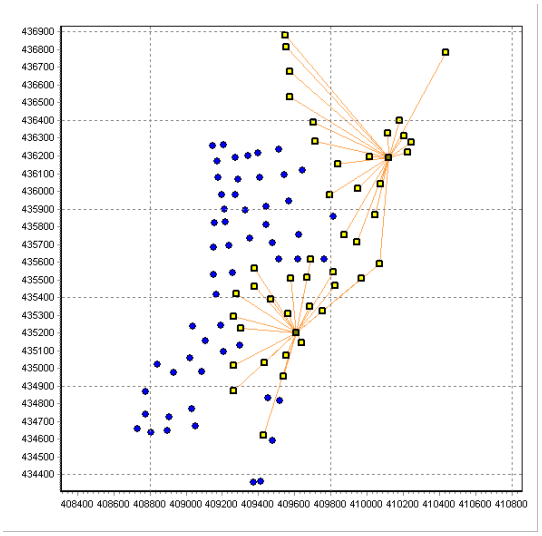

Figura 4b – Representação da redução R1b.

Figura 4 – Diferença entre os métodos de redução R1a e R1b.

Em resumo, R1a calcula quais pontos podem ser associados a uma dada mediana e R1b calcula a quais medianas um dado ponto pode ser associado.

3.3.3 Redução Medianas candidatas (R2)

Outra estratégia de redução de variáveis é eliminar a possibilidade de determinados pontos serem medianas. Embora esta estratégia implique mais facilmente na perda da possibilidade de encontrar a solução ótima do problema, ela permite que muitas variáveis sejam eliminadas, uma vez que a eliminação de um determinado nó como candidato a mediana implica na eliminação de outras *n* variáveis do modelo.

Estratégias de eliminação de medianas são mais eficientes quanto maior for a relação *n*/*p*, pois neste caso a possibilidade de eliminar medianas que compõem a solução ótima é menor.

Dado um conjunto diverso com *M* soluções, esta redução analisa quais pontos foram escolhidos pelo menos uma vez como mediana. Assim, todo ponto não selecionado como mediana nas *M* soluções avaliadas têm as variáveis que o indicam como mediana excluídas.

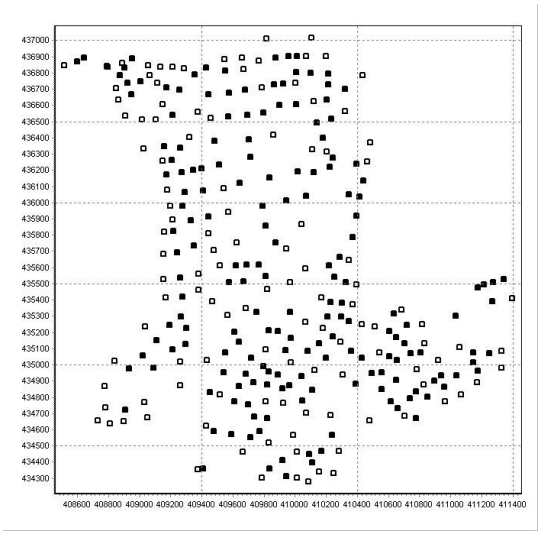

Figura 5a – Em destaque, candidatos a medianas calculados segundo a redução R2.

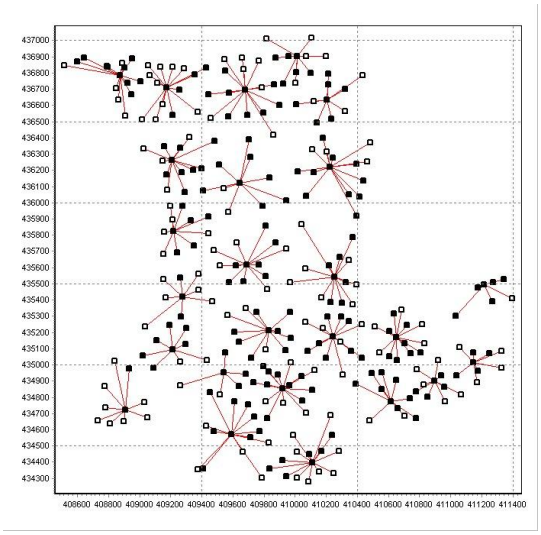

Figura 5b – Solução ótima da instância sjc3a.

Figura 5 – Representação da redução R2 e solução ótima para a instância sjc3a.

A Figura 5 traz um exemplo com a instância sjc3a (Lorena e Senne, 2004), na qual 40% das medianas foram eliminadas. Na Figura 5a, os pontos representados por pequenos quadrados de interior branco indicam que este ponto não foi escolhido como mediana em nenhuma das *M* soluções e, portanto, foi eliminada a possibilidade da mesma ser mediana. Já

os quadrados totalmente pretos indicam que o ponto foi em pelo menos uma solução escolhido como mediana, assim o mesmo mantém a possibilidade de ser mediana. A Figura 5b é semelhante a Figura 5a, mas a mesma traz a solução ótima do problema representado por linhas ligando os nós as suas respectivas medianas, mostrando que, neste caso, todas as medianas da solução ótima não foram excluídas na redução.

3.3.4 Redução a subconjuntos (R3)

Após algumas análises gráficas das diferenças entre soluções encontradas durante o processo de resolução dos modelos com o otimizador comercial, foi possível observar que quando há uma melhora do valor de função objetivo, a maioria dos agrupamentos se mantém inalterados, principalmente em instâncias cuja relação n/p é menor. Este fato sugere a possibilidade de reduzir o tamanho do problema a partir da exploração iterativa de sub-regiões com maior probabilidade de proporcionar melhora no valor da função objetivo.

O objetivo desta redução é definir uma região contendo alguns agrupamentos onde possa haver trocas entre si e, ao mesmo tempo, manter outros agrupamentos fixos, permitindo que parte da solução seja explorada mais detalhadamente.

Para definir a região a ser explorada definimos um agrupamento base, que servirá como referencial. A partir disso, definimos uma quantidade de agrupamentos para os quais são permitidas trocas de pontos entre si (agrupamentos não fixos). Esta quantidade de agrupamentos livres deve ser um valor grande o suficiente para explorar uma boa região, mas ao mesmo tempo seja pequeno o suficiente para que o otimizador tenha condições de encontrar as soluções ou provar que naquele subconjunto de pontos não há movimentos que melhorem o valor de função objetivo. Este é um parâmetro relativamente difícil de ser estabelecido de forma genérica, pois envolve capacidade do otimizador, capacidade de memória da máquina, tempo disponível para processamento, além de algumas características da instância como a relação n/p e os próprios números de pontos e medianas. No entanto, esta redução pode ser combinada com todos os outros métodos de redução, como pode ser observado na Figura 6b, na qual o número de variáveis, representadas por linhas, é significativamente reduzido com relação a Figura 6a, que possui todas as variáveis que representam possíveis ligações entre os pontos do subconjunto.

Esta estratégia mostras-se promissora, especialmente quando já se tem uma boa solução e a relação *n*/*p* é baixa, pois nestes casos a mudança na solução quando há uma melhora, em geral, afeta poucos agrupamentos.

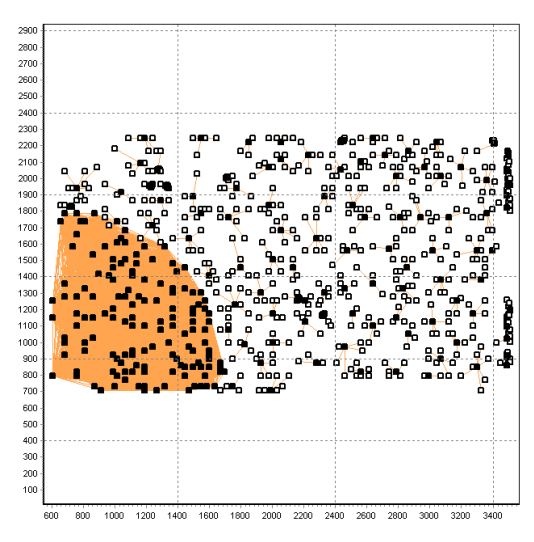

Figura 6a – Redução R3.

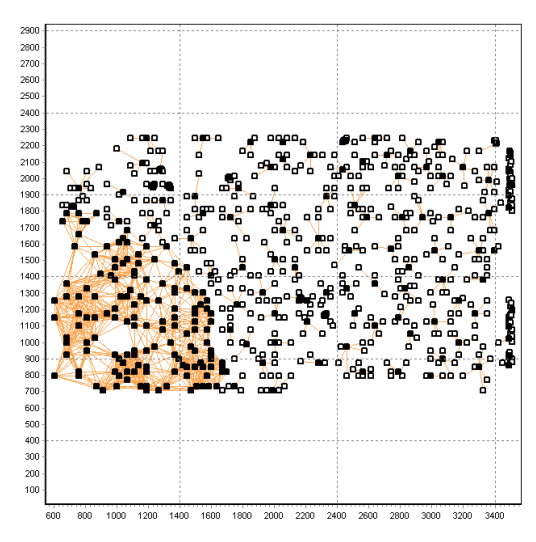

Figura 6b – Reduções R1 e R3 combinadas.

Figura 6 – Diferentes configurações da redução R3.

Neste método, dado uma mediana que representa um agrupamento de referência, são analisados os *nc* agrupamentos mais próximos desta mediana de referência. Para definir o valor de *nc*, partiu-se da observação de que o otimizador tem bom desempenho na resolução de instâncias com uma média de até 400 pontos, dependendo de certas características da instância. Deste modo, o número de agrupamentos *nc* é definido como $nc = \max(\lfloor N_{ds} * p / n \rfloor, C_{lm})$, em que N_{ds} é o número de pontos que o otimizador consegue resolver e C_{lm} é o número mínimo de agrupamentos a serem analisados, que por padrão é utilizado 5. Quando *nc* é definido como C_{lm}, o número de nós a serem analisados pode ser grande, mas a utilização combinada com outros métodos de redução tendem a minimizar este problema.

3.3.5 Redução de vizinhança de medianas (R4)

Esta estratégia parte do princípio que uma pequena perturbação em uma solução é suficiente para que o método de redução de variáveis identifique os nós que devem ser eliminados na escolha das medianas. Neste caso, a redução parte de uma solução inicial e permite a criação de variáveis que definam um ponto como mediana apenas em regiões próximas às medianas atuais.

Esta estratégia de redução analisa em cada agrupamento, os β pontos mais próximos da mediana (incluindo a mesma), com $\beta = \min(\beta; C_{ln})$, em que C_{ln} é a quantidade de nós do agrupamento analisado e β é um parâmetro inteiro positivo que define a abrangência do método de redução.

Para o caso em que $\beta = 1$, o problema restringe-se a um problema de designação, no qual há um conjunto fixo de p medianas e pode ser resolvido eficientemente com o otimizador.

A Figura 7 traz a representação de uma solução da instância u724_010, proposta e descrita na seção 3.4, na qual os pontos representados por pequenos círculos com preenchimento correspondem a pontos que mantém a possibilidade de serem medianas após a aplicação da redução (caso para $\beta = 10$). Os pontos cujos quadrados possuem o interior sem preenchimento correspondem àqueles que não podem ser medianas.

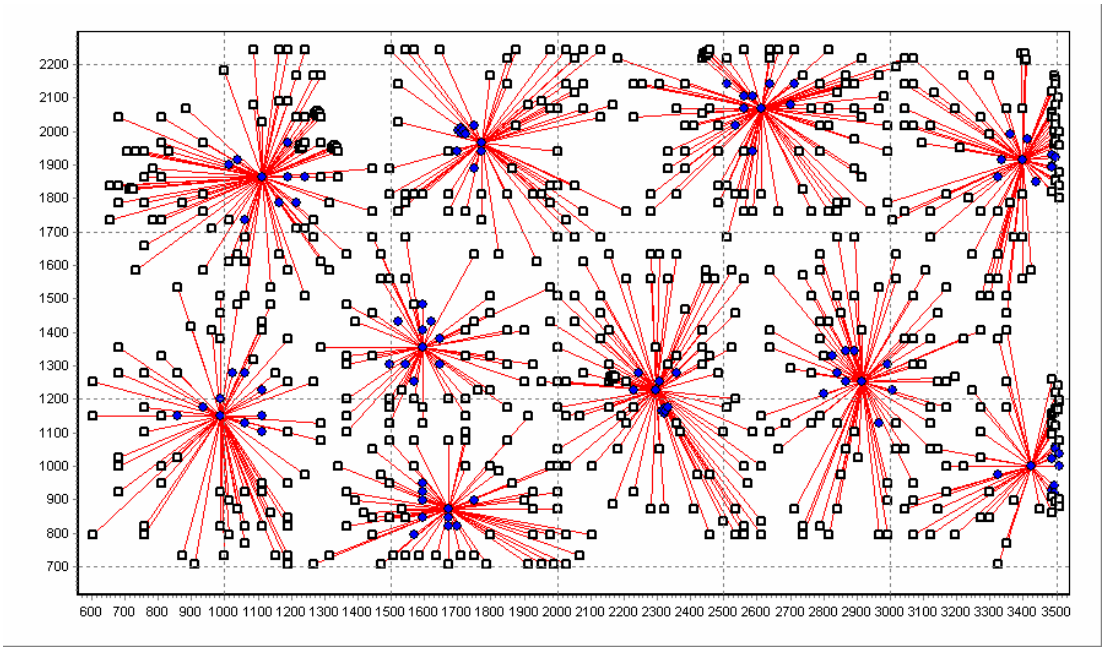

Figura 7 – Representação da redução R4.

Assim como a redução R2, esta redução possui melhor eficiência em instâncias com alta relação *n*/*p*, e sua estrutura permite que a resolução seja iterada até que não haja melhoras na função objetivo.

3.3.6 Algoritmo iterativo baseado em reduções

A utilização de algumas estratégias de redução de variáveis na resolução do modelo reduzido necessita de uma solução inicial para ajudar a definir as variáveis que serão eliminadas. Nestes casos, quando há uma melhora no valor de função objetivo, uma nova configuração de redução é formada, possibilitando que uma vizinhança diferente seja explorada e por consequência, novas soluções que permitam melhoras no valor da função objetivo sejam encontradas. É o caso das reduções R3 e R4, em especial.

O caráter deste processo iterativo juntamente com as características de cada um dos processos de redução descritos anteriormente proporcionou a criação de um algoritmo iterativo de resolução do PPMC, no qual a redução R3 é utilizada como base para explorar todo o conjunto de pontos da instância, utilizando simultaneamente as reduções R1 e R4.

A referência de um agrupamento base na redução R3 permite que sejam utilizadas diferentes estratégias para a exploração de regiões, como exemplo, apenas regiões que possuem agrupamentos com pontos cujo custo é acima da média da instância, ou que a capacidade da mediana esteja próxima ao limite de atendimento. Também é possível utilizar todos os agrupamentos como agrupamentos de referência ou aplicar uma estratégia heurística para determinar as regiões para exploração.

Neste trabalho, partindo de uma solução inicial, a seguinte heurística é proposta para determinar os agrupamentos a serem explorados de modo a analisar a instância como um todo pelo algoritmo iterativo:

Passo 1: Escolher um agrupamento *p* de referência (por padrão é escolhido um agrupamento aleatório);

Passo 2: Marcar todos os agrupamentos como não otimizados;

Passo 3: Encontrar a mediana *m* mais próxima de *p* cujo agrupamento não foi marcado como otimizado;

Passo 4: Utilizando a redução R3, explorar o subconjunto de *nc* agrupamentos mais próximos de *m*, independente se algum dos *nc* agrupamentos está marcado como

otimizado.

Passo 5: Caso não haja melhora na função objetivo da solução, marcar os agrupamentos analisados como otimizados. Caso contrário, marcar os *nc* * *h* agrupamentos mais próximos da mediana de referência como não otimizados (por padrão, *h* = 2).

Passo 6: Caso ainda existam agrupamentos não otimizados, retornar ao Passo 3;

Passo 7: Caso haja melhora no valor de função objetivo, retornar ao Passo 1 senão finaliza o algoritmo.

Para agilizar o processo de busca de uma solução factível pelo otimizador, é fornecida a solução corrente, garantindo que no pior caso não haja melhora na solução. No passo 4 é possível combinar a redução R3 com as reduções R1 e R4. O Passo 7 garante que a solução encontrada pelo algoritmo é um ótimo local para a vizinhança considerada.

Devido a especificidades das instâncias, definição de parâmetros e as características de resolução do otimizador, o tempo para exploração no Passo 4 pode variar significativamente, podendo ser tão difícil ou demorado quanto a resolução do problema original. Para contornar essa situação é definido o parâmetro T_{max} que delimita o tempo máximo de exploração no Passo 4, mesmo que este cause a perda da garantia de no final do processo iterativo obtermos um ótimo local. Para o problema reduzido, observa-se que há uma facilidade maior do otimizador em encontrar melhoras na solução em tempos menores mesmo que não consiga provar a otimalidade da solução no subconjunto analisado.

3.4 Resultados computacionais

Os testes computacionais foram realizados em um computador com processador Intel Quad-Core Xeon X3360 2.83 GHz e como otimizador foi utilizado CPLEX 12.1 com a configuração padrão.

Foram utilizados cinco conjuntos de instâncias. O primeiro (conjunto 1) é um conjunto de instâncias clássico, fornecido por Osman e Christofides (1994) e referenciadas por cpmp01 a cpmp20[5]. Neste conjunto, as primeiras 10 instâncias possuem 50 nós que devem ser agrupados em 5 medianas. Já as outras 10 instâncias restantes contêm 100 nós para 10

[5] Disponíveis em http://people.brunel.ac.uk/~mastjjb/jeb/info.html

medianas. Vale observar que para este conjunto a parte decimal das distâncias é desconsiderada.

O conjunto 2, proposto por Lorena e Senne (2004) [6], compreende seis instâncias reais obtidas de uma base de dados geográficos da cidade de São José dos Campos-SP e denominadas sjc1 a sjc4b.

O conjunto 3 compreende 5 instâncias de grande porte propostas em Lorena et al. (2003) denominadas p3038_600 a p3038_1000, onde as mesmas foram criadas a partir de uma instância com 3038 nós da TSP-LIB[7] e adaptada para incluir 600, 700, 800, 900 e 1000 agrupamentos.

Foram utilizadas ainda quatro instâncias descritas em Diaz e Fernandez (2006), contendo 737 pontos representando cidades da Espanha para 74 e 148 medianas e dois diferentes vetores de demanda. Este é considerado o conjunto 4 e a nomenclatura da instância, segundo o exemplo "spain737_74_1", indica o número de nós representando as 737 cidades espanholas; o número 74 indica o número de medianas e 1 indica o vetor de demandas utilizado.

Um novo conjunto de instâncias (conjunto 5) é proposto neste trabalho. Na sua maioria, as instâncias não podem ser resolvidas na otimalidade considerando o estado da arte atual dos otimizadores comerciais, o que justifica o uso de outra técnica de resolução já que as instâncias clássicas para este problema mostram-se de fácil resolução com a tecnologia atual. Além disso, a ideia é disponibilizar um conjunto de dados mais diverso que abrange diferentes características. Assim como em Lorena et al. (2003), este conjunto de instâncias foi adaptado das instâncias da TSP-LIB. Para cada instância foi gerado um subconjunto de cinco novas instâncias com diferentes números de medianas requeridas, de forma que contemple diferentes relações *n*/*p*. A cada ponto foi adicionada uma demanda com valor aleatório inteiro escolhido no intervalo [1; 100]. A mesma capacidade é atribuída a todas as medianas e calculada pela expressão $Q_j = \left\lceil \sum_{i \in N} q_i / (p * r) \right\rceil, j \in M$, em que r pertence ao intervalo [0,8; 0,9].

Todos os conjuntos de instâncias e suas soluções estão referenciadas ou disponíveis em http://www.inf.ufsm.br/~stefanello/instances/.

---

[6] Disponíveis em http://www.lac.inpe.br/~lorena/instancias.html
[7] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

A realização de testes iniciais com a redução R4 demonstrou que $\beta = 10$ proporciona uma boa relação entre quantidade de variáveis eliminadas e capacidade de encontrar novas soluções, sendo assumido como valor padrão. Ainda, propõe-se que esta redução seja automaticamente utilizada quando a relação $n/p > 10$ for satisfeita.

3.4.1 Resultados comparativos entre modelo forte e modelo fraco

O primeiro teste computacional foi realizado para avaliar o comportamento atual do CPLEX com a utilização da restrição (4) do modelo matemático. A Tabela 1 mostra os resultados obtidos para o conjunto 2 de instâncias com o modelo completo, onde as três primeiras colunas trazem o nome da instância, o número de nós e medianas, respectivamente. A quarta coluna, denominada FO, traz o valor de função objetivo da solução ótima. A quinta e a sexta colunas trazem respectivamente os tempos computacionais do CPLEX para resolver o modelo matemático de cada instância sem e com a utilização da restrição (4), ou seja, o modelo fraco e modelo forte. A última coluna traz o percentual do ganho relativo ao tempo computacional devido a utilização da restrição (4), definido pela fórmula $GT = (MFr - MFo) / MFo * 100$, em que *MFo* é o tempo computacional para resolução do modelo forte e *MFr* é o tempo computacional para resolução com modelo fraco.

Tabela 1 – Diferença nos tempos computacionais entre modelo forte e modelo fraco.

Instância	*n*	*p*	FO	Modelo Fraco	Modelo Forte	GT (%)
sjc1	100	10	17288,99	10,19	4,89	108,38
sjc2	200	15	33270,94	53,81	11,46	369,55
sjc3a	300	25	45335,16	134,41	62,20	116,09
sjc3b	300	30	40635,90	110,75	16,14	586,18
sjc4a	402	30	61925,51	642,34	215,60	197,93
sjc4b	402	40	52458,02	260,84	35,90	626,57

Como esperado, os resultados mostram que a utilização da restrição (4) melhora o desempenho do CPLEX para a resolução do modelo matemático. Isto se deve ao fato do método de resolução implementado no otimizador ter como base a resolução iterativa de modelos com a restrição de domínio binário relaxada, e para o problema relaxado a referida

restrição implica em soluções mais próximas da solução ótima do modelo original, facilitando a resolução. Com estes resultados fica justificado o uso do modelo forte no modelo completo para todos os testes posteriores.

3.4.2 Resultados para os conjuntos 1 e 2 de instâncias

O teste descrito nesta seção foi realizado para os conjuntos 1 e 2 de instâncias de modo a avaliar o desempenho e a capacidade de obtenção de boas soluções utilizando-se exclusivamente o método de redução R1a, pois as instâncias destes conjuntos podem ser consideradas de pequeno porte. Nestes testes, não foi possível observar uma dominância estrita entre as reduções R1a e R1b.

A Tabela 2 e a Tabela 3 trazem os resultados para os conjuntos 1 e 2 de instâncias, na qual a primeira coluna traz o nome da instância, a coluna com cabeçalho Tempo(s) / FO traz o tempo computacional para resolução do modelo completo/ forte e a solução ótima do problema. Estes valores servem como referência dos dados das colunas seguintes. A coluna Dados indica se os valores das próximas colunas referem-se ao tempo computacional ou ao valor de função objetivo obtido com diferentes valores de α. Os dados relativos ao ganho em tempo computacional são calculados pela fórmula $GT = (T_c - T_\alpha) / \min(T_c, T_\alpha) * 100$, em que T_c é o tempo computacional para resolução do modelo contendo todas as variáveis (modelo completo), e T_α é o tempo computacional para resolução utilizando um fator de redução α. A linha referente ao termo *gap* indica a folga do limitante superior, calculado utilizando a redução R1a, em relação ao valor ótimo da instância (afim de uma melhor visualização, os valores nulos são omitidos).

Tabela 2 – Resultados de diferentes valores de α para o conjunto 1 de instâncias.

			Fator de redução (α)							
Inst	**Tempo(s)/ FO**	**Dados**	**1,0**	**1,2**	**1,4**	**1,6**	**1,8**	**2,0**	**2,2**	**2,4**
cpmp01	0,25	GT (%)	1.150,00	92,31	127,27	92,31	19,05	-4,00	56,25	25,00
	713	*gap* (%)	5,19	0,56	–	–	–	–	–	–
cpmp02	0,10	GT (%)	233,33	400,00	400,00	400,00	66,67	233,33	150,00	25,00
	740	*gap* (%)	12,57	1,08	1,08	0,14	–	–	–	–
cpmp03	0,23	GT (%)	1.050,00	130,00	15,00	27,78	43,75	91,67	76,92	27,78
	751	*gap* (%)	5,46	0,27	0,27	0,27	–	–	–	–
cpmp04	0,12	GT (%)	500,00	140,00	200,00	300,00	200,00	100,00	100,00	100,00
	651	*gap* (%)	8,14	4,76	3,69	2,30	–	–	–	–
cpmp05	0,32	GT (%)	1.500,00	128,57	966,67	100,00	190,91	166,67	146,15	77,78
	664	*gap* (%)	9,19	8,58	1,51	1,20	1,20	–	–	–
cpmp06	0,16	GT (%)	220,00	0,00	14,29	-6,25	100,00	77,78	60,00	128,57
	778	*gap* (%)	7,71	4,24	1,41	1,41	–	–	–	–
cpmp07	0,45	GT (%)	1.400,00	800,00	181,25	66,67	-126,67	-97,78	15,38	50,00
	787	*gap* (%)	29,48	9,40	7,62	3,43	2,29	2,29	0,25	0,25
cpmp08	10,43	GT (%)	34.666,67	1.867,92	1.555,56	2.443,90	1.555,56	98,29	102,92	121,91
	820	*gap* (%)	infes	5,61	0,85	0,12	0,12	–	–	–
cpmp09	0,41	GT (%)	1.266,67	64,00	115,79	51,85	24,24	156,25	173,33	24,24
	715	*gap* (%)	infes	1,96	0,84	0,56	0,56	–	–	–
cpmp10	2,51	GT (%)	6.175,00	392,16	198,81	-50,20	274,63	114,53	175,82	94,57
	829	*gap* (%)	7,36	4,83	3,74	3,74	0,72	0,72	–	–
cpmp11	2,24	GT (%)	65,93	300,00	124,00	239,39	121,78	32,54	37,42	105,50
	1006	*gap* (%)	9,64	0,40	0,10	0,00	–	–	–	–
cpmp12	2,25	GT (%)	1.400,00	508,11	20,97	17,80	42,41	196,05	102,70	55,17
	966	*gap* (%)	8,70	4,04	2,28	0,72	–	–	–	–
cpmp13	0,98	GT (%)	476,47	216,13	188,24	139,02	157,89	104,17	104,17	68,97
	1026	*gap* (%)	10,92	1,17	0,49	0,49	0,10	0,10	–	–
cpmp14	11,57	GT (%)	7.613,33	627,67	539,23	623,13	588,69	85,71	32,53	66,47
	982	*gap* (%)	11,10	6,31	1,02	0,10	0,10	0,10	0,10	0,10
cpmp15	11,73	GT (%)	7.231,25	795,42	624,07	392,86	416,74	356,42	321,94	57,66
	1091	*gap* (%)	7,24	1,47	0,73	0,09	–	–	–	–
cpmp16	2,55	GT (%)	4.150,00	240,00	339,66	325,00	571,05	183,33	183,33	109,02
	954	*gap* (%)	7,02	4,72	3,67	1,36	0,31	0,31	0,10	0,10
cpmp17	9,42	GT (%)	13.357,14	41,02	748,65	435,23	223,71	281,38	352,88	266,54
	1034	*gap* (%)	3,00	2,32	0,19	0,19	–	–	–	–
cpmp18	3,86	GT (%)	1.508,33	-66,32	136,81	71,56	77,88	75,45	82,94	104,23
	1043	*gap* (%)	4,03	1,34	–	–	–	–	–	–
cpmp19	4,28	GT (%)	370,33	169,18	355,32	87,72	90,22	41,72	40,33	59,70
	1031	*gap* (%)	12,22	2,81	1,16	0,97	0,68	0,68	0,68	–
cpmp20	72,64	GT (%)	38.131,58	4.275,90	32,89	67,37	46,48	12,83	-9,29	26,59
	1005	*gap* (%)	infes	2,89	2,19	1,29	0,70	0,60	0,20	–
	GT médio		**2.847,12***	**556,10**	**344,22**	**291,26**	**234,25**	**115,32**	**115,29**	**79,74**
	***gap* médio**		**9,35***	**3,44**	**1,64**	**0,92**	**0,34**	**0,24**	**0,07**	**0,02**

*média calculada desconsiderando os resultados para os quais não foi possível encontrar solução factível

Tabela 3 – Resultados de diferentes valores de α para o conjunto 2 de instâncias.

			Fator de redução (α)							
Inst	**Tempo(s)/ FO**	**Dados**	**1,0**	**1,2**	**1,4**	**1,6**	**1,8**	**2,0**	**2,2**	**2,4**
sjc1	4,89	GT (%)	398,98	307,50	246,81	207,55	154,69	221,71	81,78	60,86
	17288,99	*gap* (%)	5,82	2,25	0,99	0,72	–	–	–	–
sjc2	11,46	GT (%)	991,43	481,73	362,10	227,43	126,48	141,77	167,76	208,06
	33270,94	*gap* (%)	0,65	0,10	0,07	0,07	0,07	–	–	–
sjc3a	62,20	GT (%)	650,30	1.849,84	760,30	351,38	174,49	129,44	323,13	96,28
	45335,16	*gap* (%)	0,95	0,48	–	–	–	–	–	–
sjc3b	16,14	GT (%)	495,57	443,43	502,24	723,47	418,97	337,40	553,44	537,94
	40635,90	*gap* (%)	0,45	0,39	0,07	0,07	0,07	0,07	–	–
sjc4a	215,60	GT (%)	7.086,67	4.152,47	1.612,47	1.191,79	568,11	166,77	218,98	181,61
	61925,51	*gap* (%)	1,44	0,47	0,30	0,08	–	–	–	–
sjc4b	35,90	GT (%)	609,49	626,72	413,59	436,62	392,46	424,09	409,94	310,29
	52458,02	*gap* (%)	0,57	0,26	0,23	0,01	–	–	–	–
	GT médio		**1.705,41**	**1.310,28**	**649,58**	**523,04**	**305,87**	**236,86**	**292,51**	**232,51**
	***gap* médio**		**1,65**	**0,66**	**0,28**	**0,16**	**0,02**	**0,01**	**–**	**–**

De acordo com os dados tabelados, é possível observar que mesmo para valores pequenos de α é possível obter soluções de boa qualidade em tempos computacionais significativamente menores se comparados ao tempo de resolução do modelo completo. Deve-se observar que ocorrem casos em que o tempo computacional para resolver o modelo reduzido é superior ao modelo completo (valores negativos), no entanto, na média, é possível inferir uma significativa queda no tempo computacional decorrente das reduções.

Para o conjunto 2 de instâncias (sjc1 a sjc4a) foi possível encontrar todas as soluções ótimas com o modelo reduzido para $\alpha > 2$. Já para o conjunto 1 (cpmp01 a cpmp20), embora não tabulado, para $\alpha = 2{,}6$ foi possível obter todas as soluções ótimas, com exceção da instância cpmp16, caso em que a solução ótima foi obtida para $\alpha = 3$.

Observa-se também que a utilização do fator α muito reduzido, embora implique numa significativa redução de tempo, causa uma maior degeneração na solução, inclusive causando a infactibilidade do problema, como nos casos cpmp08, cpmp09 e cpmp20 para $\alpha = 1$, identificados na tabela como "infes". De tudo, pode-se afirmar que $\alpha = 2$ traz uma boa relação entre redução do tempo computacional e qualidade de solução obtida.

Apesar da redução R1a se mostrar eficiente neste conjunto, em algumas instâncias para as quais há pontos muito dispersos ou distantes de outros ou que possuem uma demanda bem acima da demanda média, ou seja, em instâncias menos estruturadas, esta redução pode causar uma maior degeneração na qualidade da solução final.

A Tabela 4 traz os melhores resultados da literatura dos últimos anos para o conjunto 2 de instâncias. A primeira coluna traz a identificação da instância, as colunas FO e Tempo trazem respectivamente o valor de função objetivo e o tempo computacional das referidas abordagens citadas no cabeçalho da tabela. A primeira abordagem (SS) é uma metaheurística híbrida que combina *Scatter Search* com Religamento de Caminhos proposta em Scheuerer e Wendolsky (2006). Já a abordagem VNS é descrita em Fleszar e Hindi (2008) e usa metaheurística VNS combinada com CPLEX, a terceira abordagem, descrita em Chaves et al. (2007) e denominada CS, é uma heurística híbrida que consiste na detecção de regiões promissoras baseada em agrupamentos. Em Fen-Cplex (Boccia et al., 2008) os autores propõem um algoritmo de planos de corte baseado em cortes de Fenchel. Por último, são mostrados os dados obtidos utilizando o modelo reduzido utilizando apenas a redução R1a com $\alpha = 2{,}2$.

Tabela 4 – Resultados do modelo reduzido comparado com os resultados da literatura.

	SS[a]		VNS[b]		CS[c]		Fen-Cplex[d]		Mod. Reduzido[e] (α=2.2)	
ID	**FO**	**Tempo**	**FO**	**Tempo**	**FO**	**Tempo**	**FO**	**Tempo**	**FO**	**Tempo**
sjc1	**17.288,99**	60,00	**17.288,99**	50,50	**17.288,99**	22,72	**17.288,99**	37,60	**17.288,99**	2,69
sjc2	33.293,40	600,00	**33.270,94**	44,08	**33.270,94**	112,81	**33.270,94**	127,90	**33.270,94**	4,28
sjc3a	45.338,02	2.307,00	**45.335,16**	8.580,30	**45.335,16**	940,75	**45.335,16**	495,10	**45.335,16**	14,70
sjc3b	**40.635,90**	2.308,00	**40.635,90**	2.292,86	**40.635,90**	1.887,97	**40.635,90**	72,20	**40.635,90**	2,47
sjc4a	**61.925,52**	6.109,00	**61.925,51**	4.221,47	61.928,72	2.885,11	**61.925,51**	1.209,50	**61.925,51**	67,59
sjc4b	52.531,46	6.106,00	52.469,96	3.471,44	52.531,27	7.626,33	**52.458,00**	669,70	**52.458,02**	7,04

[a] Intel Celeron 2.2 GHz
[b] Pentium IV 3.2 GHz e CPLEX versão 10.1
[c] Pentium IV 3.02 GHz
[d] ASUS S5 notebook 1.6 GHz e CPLEX versão 9.0
[e] Intel Quad-Core Xeon X3360 2.83 GHz e CPLEX versão 12.1

A tabela mostra que, apesar das diferentes plataformas de *hardware*, o tempo computacional do modelo reduzido é significativamente menor que os demais resultados. Ainda, com a utilização do fator de redução $\alpha = 2{,}2$ o CPLEX conseguiu encontrar as soluções ótimas para todas as instâncias do conjunto.

3.4.3 Resultados para o conjunto 3 de instâncias

Para o terceiro conjunto de instâncias os testes iniciais mostraram que a redução R1b é superior a redução R1a, de forma que os resultados para estas instâncias e as seguintes são calculados a partir da redução R1b. Diferentes valores para o parâmetro α foram testados, os quais são reportados na Tabela 5. A primeira coluna traz o nome da instância, e as demais colunas trazem os valores de função objetivo para diferentes valores do fator de redução α para um tempo limite de execução de 2000 segundos.

Tabela 5 – Resultados computacionais para diferentes valores de α.

	Fator de redução (α)				
Instância	**2**	**2,5**	**3**	**3,5**	**4**
p3038_600	122.934,55	**122.785,83**	122.935,74	122.857,29	124.293,37
p3038_700	109.812,28	**109.804,82**	109.817,18	109.831,35	109.953,54
p3038_800	100.148,33	**100.114,85**	100.195,68	100.154,88	100.188,83
p3038_900	**92.339,47**	92.347,90	92.348,26	92.360,63	92.362,69
p3038_1000	85.872,48	85.859,52	**85.856,77**	85.869,55	85.882,47

Os melhores resultados são destacados em negrito, e mostram que, dada a limitação do tempo, o CPLEX apresentou desempenho superior com $\alpha = 2,5$.

O desempenho do CPLEX no decorrer do tempo do teste anterior para $\alpha = 2,5$ é mostrado na Figura 8, em que a Figura 8a mostra o desempenho para a instância p3038_900 e a Figura 8b para a instância p3038_1000, instâncias com 3038 nós e 900 e 1000 medianas respectivamente. Na figura, os pontos representam algumas das soluções encontradas pelo CPLEX que melhoram o valor da função objetivo, e a linha representa a evolução da resolução no decorrer do tempo.

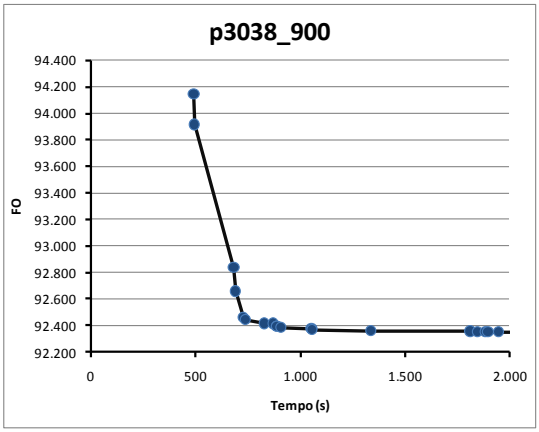

Figura 8a – Representação do valor da F.O obtida no decorrer do tempo para a instância p3038_900

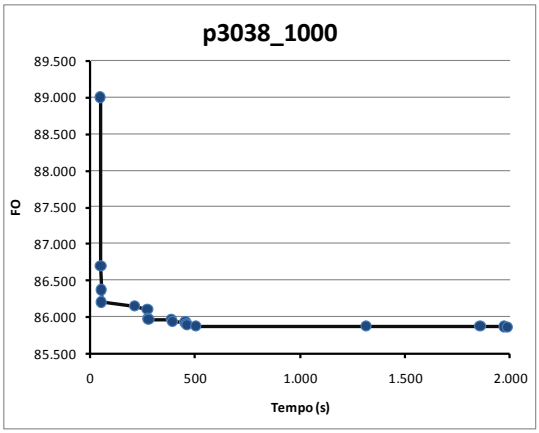

Figura 8b – Representação do valor da F.O obtida no decorrer do tempo para a instância p3038_1000

Figura 8 – Representação do valor da F.O obtida no decorrer do tempo.

É possível observar na Figura 8 que mesmo na situação onde o CPLEX demora mais para encontrar uma solução inicial (Figura 8a), depois da solução obtida, ele é capaz de melhorá-la rapidamente até patamares próximos às melhores soluções conhecidas possibilitando ainda seguidas melhoras no decorrer do tempo. Já na Figura 8b é possível observar que num curto espaço de tempo o CPLEX já foi capaz de encontrar uma solução e melhorá-la significativamente nos passos seguintes.

A solução encontrada com o modelo reduzido para α = 2,5 foi utilizada como solução inicial dos testes realizados com o algoritmo iterativo fazendo uso das reduções R1b e R3, com parâmetros α = 4, de modo a proporcionar uma exploração maior da vizinhança, e N_{ds} = 350. Além disso, o tempo de cada exploração de um determinado subconjunto (T_{max}) foi limitado a 100 segundos.

Dado que o algoritmo iterativo possui um caráter aleatório na escolha do agrupamento de referência, para obtenção de resultados mais precisos foram realizadas 10 execuções do procedimento. Os resultados são apresentados na Tabela 6, na qual as primeiras colunas trazem respectivamente o nome da instância, o número de nós e de medianas. Em Lorena e Senne (2004) os autores usam geração de colunas para calcular o *lower bound* das instâncias deste conjunto. Os resultados são reportados nas colunas Limitante inferior e Tempo Total que respectivamente trazem o valor do *lower bound* e o tempo computacional em segundos. As colunas seguintes referem-se aos dados obtidos com as 10 execuções do algoritmo iterativo. As colunas Min, Med e Max trazem os valores de função objetivo mínimo, médio e máximo. A coluna DP traz o desvio padrão das soluções obtidas e a coluna Tempo Med

reporta o tempo médio em segundos das 10 execuções. Observa-se que no tempo médio reportado na última coluna não está incluso o tempo para a obtenção da solução inicial fornecida ao algoritmo iterativo.

Tabela 6 – Resultados computacionais do Algoritmo Iterativo para o conjunto 3.

			Geração de colunas[f]		Algoritmo Iterativo[g]				
Instância	***n***	***p***	**Limitante inferior**	**Tempo Total (s)**	**Min**	**Med**	**Max**	**DP**	**Tempo Med.**
p3038_600	3038	600	122.020,66	59.593.02	122.720,86	122.725,95	122.739,98	6,85	2.187,30
p3038_700	3038	700	108.685,59	46.705.53	109.672,37	109.674,25	109.681,53	3,84	3.322,63
p3038_800	3038	800	98.530,99	33.844.27	100.067,30	100.068,60	100.073,56	2,50	3.027,81
p3038_900	3038	900	90.239,65	25.306.54	92.310,63	92.312,94	92.322,68	3,81	1.931,35
p3038_1000	3038	1000	83.231,58	20.210.25	85.854,66	85.855,51	85.856,26	0,63	1.042,24

[f] Sun Ultra30 workstation
[g] Intel Quad-Core Xeon X3360 2.83 GHz e CPLEX versão 12.1

No caso das instâncias com 700 e 800 medianas, a solução com menor valor de função objetivo foi encontrada em sete das dez iterações, três na de 600, duas na de 1000 e em uma iteração na instância com 900 medianas. Nas piores soluções a frequência foi única para as instâncias com 600, 800 e 900 nós, dupla para a de 700 e tripla para a instância com 1000 medianas. Observa-se ainda que a média do valor de função objetivo das soluções encontradas fica muito próxima as melhores soluções, mantendo valores de desvio padrão relativamente baixos e um *gap* (valor que representa o quão distante a solução encontrada está da melhor solução conhecida) praticamente nulo. Tais resultados indicam que a aleatoriedade no algoritmo iterativo não é um fator de suma importância na resolução, em especial para este conjunto de instâncias.

Já em referência ao tempo computacional médio utilizado pelo algoritmo iterativo, em comparação ao tempo necessário para obtenção dos limitantes pelo método de geração de colunas reportado em Lorena e Senne (2004), a estrutura de resolução mostrou-se eficiente, embora os tempos sejam reportados para diferentes configurações de *hardware*.

No comparativo das melhores soluções encontradas pelo modelo reduzido e o limitante inferior, o *gap* relativo varia de 0,57% para a instância p3038_600 até 3,15% para a instância p3038_1000. No entanto não é possível afirmar se esta diferença é devido aos valores do limitante inferior ou dos obtidos com o algoritmo iterativo.

As melhoras nos valores de função objetivo obtidas pelo algoritmo iterativo em comparação ao obtido com o modelo reduzido apenas utilizando a redução R1 não chegam ser notavelmente significativas para este conjunto de instâncias. Muito deste fato é atribuído a boa qualidade da solução obtida com o modelo reduzido, visto que a baixa relação *n*/*p* favorece a resolução do mesmo. Neste caso, o algoritmo iterativo realizaria a função de pós-processamento caso houvesse a disponibilidade de mais tempo de processamento para tomada de decisão.

3.4.4 Resultados para o conjunto 4 de instâncias

A resolução das quatro instâncias do conjunto 4 foi dividida em duas fases. A primeira é a obtenção de uma solução inicial e a segunda a utilização do algoritmo iterativo na busca por melhores soluções. A solução inicial foi gerada a partir do modelo matemático reduzido com redução R1b (α = 3,5) e tempo máximo de busca (T_{max}) limitado em 500 segundos. Os valores da função objetivo obtidos são reportados na coluna Fase1 da Tabela 7. Partindo da solução encontrada na primeira fase, na segunda fase foi utilizado o algoritmo iterativo utilizando a redução R3 com os parâmetros N_{ds} = 250 e T_{max} = 60 segundos, juntamente com a redução R1b, ainda com α = 3,5. O valor de função objetivo da solução encontrada na segunda fase e o tempo computacional total são mostrados respectivamente nas duas últimas colunas da Tabela 7, sendo que a primeira coluna da tabela traz o nome da instância seguido dos valores de função objetivo e tempo computacional reportados por Diaz e Fernandez (2006).

Tabela 7 – Resultados computacionais para o quarto conjunto de instâncias.

	SS-PR[h]		Modelo Reduzido[i]		
Instância	**FO**	**Tempo**	**Fase1 (R1b)**	**Fase2 (R1b+R3)**	**TempoTotal**
spain737_74_1	8.967	9.717,45	9.227	8.845	3.340,30
spain737_74_2	8.970	57.239,12	9.201	8.856	3.330,88
spain737_148_1	6.012	43.361,02	5.903	5.901	765,25
spain737_148_2	6.009	17.714,15	5.920	5.917	707,00

[h] Sun Blade 1000/750
[i] Intel Quad-Core Xeon X3360 2.83 GHz e CPLEX versão 12.1

Observa-se que o tempo computacional para as instâncias com 148 medianas foi inferior as outras duas instâncias com 74 medianas. Isso se deve ao fato de que a solução inicial foi obtida tão somente com a redução R1b, a qual é mais favorável a instâncias que possuem baixa relação *n*/*p*, permitindo que o número de variáveis do modelo reduzido seja menor, com a consequente melhora no desempenho do otimizador, fornecendo de imediato soluções de melhor qualidade.

Em comparação com os resultados reportados em Diaz e Fernandez (2006), foi possível melhorar os resultados de todas as instâncias. Para as instâncias com 148 medianas, as soluções encontradas já na primeira fase foram melhores. Ainda que os dados reportados sejam oriundos de plataformas de *hardware* distintas, pode-se inferir que a redução do tempo computacional decorrente do uso do modelo reduzido é representativa.

Ao contrário dos outros conjuntos de instâncias, estas possuem uma matriz de distância não euclidiana. Observou-se nitidamente que o CPLEX teve maior dificuldade de provar a otimalidade do modelo, tanto na primeira quanto na segunda fase, mesmo utilizando um valor menor para o parâmetro N_{ds} do que nos testes dos outros conjuntos de instâncias.

3.4.5 Resultados para o conjunto 5 de instâncias

A seguir são relatados os testes com o conjunto 5 de instâncias. Dentre todos os conjuntos, este possui a maior diversidade de instâncias e, apesar da possibilidade de ajustar os parâmetros para cada instância, optou-se por configurar os parâmetros de forma única para

todo o conjunto, implicando numa configuração genérica para ser utilizada como estratégia geral de resolução.

O procedimento de resolução foi divido em três fases. A primeira é a obtenção de uma solução inicial fazendo uso do gerador diverso de soluções utilizando um critério de parada de 1000 iterações. A segunda fase parte da solução encontrada na fase anterior e utiliza o modelo reduzido com as estratégias de redução de variáveis denominadas R1b e R4 com os parâmetros $\alpha = 2$, $\beta = 10$ e $T_{max} = n$ em que n é o número de nós da instância. A terceira fase também parte da solução encontrada na fase anterior e usa algoritmo iterativo com as reduções R1b, R3 e R4 com parâmetros $\alpha = 4$ para a redução R1b, $N_{ds} = 400$ para R3, $\beta = 14$ para R4 e $T_{max} = 100$ segundos para o algoritmo iterativo. Estes parâmetros foram definidos após vários testes computacionais com diferentes configurações, e escolhidos por proporcionarem melhor relação entre qualidade de solução e tempo computacional. Os parâmetros e seus respectivos valores utilizados em cada uma das fases estão resumidos na Tabela 8.

Tabela 8 – Resumo dos parâmetros utilizados nas três fases da resolução.

Parâmetros	Fase1	Fase2	Fase3
Iterações	1000	–	–
α	–	2	4
β	–	10	14
T_{max}	–	n	100
N_{ds}	–	–	400

Devido a aleatoriedade inerente ao algoritmo de obtenção de soluções iniciais e do algoritmo iterativo, para uma melhor avaliação das estratégias de resolução propostas os testes foram repetidos 10 vezes. As Tabelas 9, 10 e 11 trazem os dados obtidos para cada uma das três fases do procedimento de resolução, onde as três primeiras colunas de cada tabela referenciam características da instância como nome, quantidade de nós e medianas, respectivamente. Já a coluna MR traz os melhores resultados dos valores de função objetivo obtidos em todos os testes. Ainda, são reportados os valores de função objetivo mínimo, médio e máximo das 10 execuções representados pelas colunas min, med e max, respectivamente. A coluna *gap* traz os valores médios do *gap* das 10 execuções de cada instância e por último é reportado o tempo médio acumulado em segundos para cada uma das

fases. O valor do *gap* é calculado pela fórmula *gap* = (FO − MR) * 100 / MR, em que MR é o valor de função objetivo da melhor solução conhecida e FO é o valor de função objetivo que se deseja avaliar.

Para este conjunto de instâncias o critério de comparação das soluções encontradas são as melhores soluções obtidas em todos os testes computacionais realizados durante a pesquisa, o que inclui as 10 execuções com os parâmetros citados anteriormente e os testes de calibragem dos parâmetros. Para calibragem dos parâmetros, a primeira fase envolveu testes com 1000 e 2000 iterações do gerador diverso de soluções. Na segunda fase foram executados testes com parâmetros $\alpha = 2$, $\beta = \{5; 6; 7; 8; 9; 10\}$ e $T_{max} = \{n; n/2\}$. Para a terceira fase as diferentes configurações de parâmetros foram com $\alpha = \{3; 3{,}5; 4\}$, $N_{ds} = \{300; 350; 400\}$, $\beta = \{8; 10; 12; 14\}$ e $T_{max} = \{30; 60; 100\}$, com todos os testes utilizando a mesma semente do gerador de números aleatórios. Ainda, em instâncias de menor porte, foi resolvido com CPLEX o modelo completo e para as instâncias que o otimizador conseguiu provar a otimalidade, os valores são reportados em negrito.

Tabela 9 – Resultados computacionais da Fase 1 para conjunto 5.

				Fase1				
Instância	***n***	***p***	**MR**	**min**	**med**	**max**	***gap***	**Tempo(s)**
lin318_005	318	5	**180.281,20**	180.886,94	187.187,02	205.390,77	3,83	2,11
lin318_015	318	15	**88.901,56**	93.686,72	95.368,73	96.650,77	7,27	1,22
lin318_040	318	40	47.988,38	50.561,77	51.960,97	53.712,21	8,28	1,22
lin318_070	318	70	**32.198,64**	34.690,83	35.965,26	37.522,78	11,70	1,61
lin318_100	318	100	**22.942,69**	26.217,21	27.510,16	28.798,88	19,91	1,87
ali535_005	535	5	**9.956,77**	9.971,43	10.245,99	11.242,67	2,90	6,09
ali535_025	535	25	**3.695,15**	3.925,03	4.165,79	4.525,59	12,74	2,34
ali535_050	535	50	2.461,41	2.711,86	2.798,52	2.891,30	13,70	2,61
ali535_100	535	100	1.438,42	1.656,39	1.738,48	1.847,88	20,86	3,76
ali535_150	535	150	1.032,28	1.261,57	1.299,63	1.340,20	25,90	4,65
u724_010	724	10	182.102,31	184.084,08	186.890,20	191.684,21	2,63	5,68
u724_030	724	30	**95.034,01**	98.150,23	99.988,17	103.139,05	5,21	3,56
u724_075	724	75	54.735,05	57.551,11	58.947,24	60.515,47	7,70	4,58
u724_125	724	125	38.976,76	41.969,04	42.527,14	44.440,02	9,11	6,18
u724_200	724	200	28.079,97	30.657,56	31.500,22	32.961,10	12,18	8,13
rl1304_010	1304	10	2.147.312,37	2.182.146,70	2.221.318,58	2.265.836,74	3,45	16,92
rl1304_050	1304	50	802.283,41	868.636,73	890.228,84	930.107,35	10,96	8,49
rl1304_100	1304	100	498.090,74	531.076,24	551.772,55	580.222,35	10,78	10,95
rl1304_200	1304	200	276.977,60	316.495,56	329.166,73	354.841,08	18,84	17,26
rl1304_300	1304	300	191.224,85	224.211,30	232.109,73	239.305,82	21,38	22,60
pr2392_020	2392	20	2.238.410,94	2.242.609,97	2.297.455,75	2.342.512,36	2,64	31,40
pr2392_075	2392	75	1.092.294,02	1.136.354,00	1.165.944,06	1.209.031,58	6,74	21,04
pr2392_150	2392	150	711.111,25	756.303,17	777.686,38	808.714,00	9,36	29,02
pr2392_300	2392	300	458.145,29	501.203,89	509.696,41	519.581,56	11,25	48,84
pr2392_500	2392	500	316.042,97	350.319,30	356.130,27	359.548,67	12,68	71,17
fnl4461_0020	4461	20	1.285.523,48	1.294.710,12	1.308.317,10	1.326.876,14	1,77	99,69
fnl4461_0100	4461	100	550.043,01	565.678,62	572.486,36	577.798,25	4,08	52,24
fnl4461_0250	4461	250	336.109,51	353.940,38	356.958,71	360.074,57	6,20	86,06
fnl4461_0500	4461	500	224.671,57	238.991,14	240.845,00	243.738,84	7,20	152,70
fnl4461_1000	4461	1000	145.862,38	158.285,85	159.480,75	161.733,43	9,34	260,69

Tabela 10 – Resultados computacionais da Fase 2 para conjunto 5.

				Fase2				
Instância	***n***	***p***	**MR**	**min**	**med**	**max**	**gap**	**Tempo(s)**
lin318_005	318	5	**180.281,20**	180.281,20	180.281,32	180.282,38	0,00	14,96
lin318_015	318	15	**88.901,56**	88.901,56	88.901,56	88.901,56	0,00	29,13
lin318_040	318	40	47.988,38	47.988,38	48.043,77	48.176,63	0,12	319,38
lin318_070	318	70	**32.198,64**	32.198,64	32.290,39	32.300,58	0,28	192,97
lin318_100	318	100	**22.942,69**	23.376,35	23.634,82	23.724,11	3,02	320,06
ali535_005	535	5	**9.956,77**	9.956,77	10.216,01	11.229,19	2,60	12,54
ali535_025	535	25	**3.695,15**	3.729,20	3.782,63	3.889,79	2,37	523,03
ali535_050	535	50	2.461,41	2.476,87	2.498,73	2.518,24	1,52	537,94
ali535_100	535	100	1.438,42	1.448,15	1.452,70	1.458,26	0,99	539,09
ali535_150	535	150	1.032,28	1.037,35	1.042,97	1.051,96	1,04	539,98
u724_010	724	10	182.102,31	182.702,70	184.522,63	187.420,01	1,33	31,46
u724_030	724	30	**95.034,01**	95.893,26	96.509,27	97.860,83	1,55	121,03
u724_075	724	75	54.735,05	54.735,05	54.736,90	54.753,50	0,00	609,80
u724_125	724	125	38.976,76	38.986,77	38.991,05	39.012,09	0,04	730,74
u724_200	724	200	28.079,97	28.101,88	28.115,59	28.121,76	0,13	727,26
rl1304_010	1304	10	2.147.312,37	2.153.190,86	2.190.110,34	2.221.924,22	1,99	92,78
rl1304_050	1304	50	802.283,41	818.034,28	826.986,40	836.835,95	3,08	1.237,70
rl1304_100	1304	100	498.090,74	500.523,86	503.125,17	505.481,31	1,01	1.316,58
rl1304_200	1304	200	276.977,60	277.063,80	277.165,16	277.290,07	0,07	1.322,82
rl1304_300	1304	300	191.224,85	191.294,52	191.340,06	191.441,47	0,06	1.328,18
pr2392_020	2392	20	2.238.410,94	2.241.089,23	2.275.651,05	2.297.039,34	1,66	279,09
pr2392_075	2392	75	1.092.294,02	1.107.647,74	1.116.559,46	1.127.337,82	2,22	1.119,14
pr2392_150	2392	150	711.111,25	716.723,15	721.017,88	726.566,72	1,39	2.426,21
pr2392_300	2392	300	458.145,29	458.275,39	458.673,44	458.913,40	0,12	2.446,04
pr2392_500	2392	500	316.042,97	316.277,57	316.526,86	316.759,21	0,15	2.468,26
fnl4461_0020	4461	20	1.285.523,48	1.290.517,87	1.301.620,94	1.315.483,54	1,25	509,58
fnl4461_0100	4461	100	550.043,01	556.679,02	560.085,49	563.644,57	1,83	4.494,37
fnl4461_0250	4461	250	336.109,51	338.906,80	342.035,94	358.611,51	1,76	4.564,25
fnl4461_0500	4461	500	224.671,57	225.006,19	225.493,13	226.720,52	0,37	4.630,75
fnl4461_1000	4461	1000	145.862,38	145.876,75	145.896,79	145.927,60	0,02	4.738,80

Tabela 11 – Resultados computacionais da Fase 3 para conjunto 5.

				Fase3				
Instância	***n***	***p***	**MR**	**min**	**med**	**max**	**gap**	**Tempo(s)**
lin318_005	318	5	**180.281,20**	180.281,20	180.281,20	180.281,20	0,00	133,85
lin318_015	318	15	**88.901,56**	88.901,56	88.901,56	88.901,56	0,00	56,92
lin318_040	318	40	47.988,38	47.988,38	48.028,97	48.076,79	0,08	469,61
lin318_070	318	70	**32.198,64**	32.198,64	32.198,64	32.198,64	0,00	365,51
lin318_100	318	100	**22.942,69**	22.942,69	22.943,45	22.950,25	0,00	660,46
ali535_005	535	5	**9.956,77**	9.956,77	10.210,58	11.225,83	2,55	68,02
ali535_025	535	25	**3.695,15**	3.695,15	3.696,98	3.703,64	0,05	2.188,37
ali535_050	535	50	2.461,41	2.472,15	2.485,16	2.497,26	0,96	1.762,26
ali535_100	535	100	1.438,42	1.441,87	1.447,60	1.454,90	0,64	2.353,84
ali535_150	535	150	1.032,28	1.032,28	1.036,26	1.041,12	0,39	2.605,05
u724_010	724	10	182.102,31	182.303,76	183.244,16	186.798,98	0,63	96,30
u724_030	724	30	**95.034,01**	95.034,01	95.118,31	95.285,83	0,09	376,00
u724_075	724	75	54.735,05	54.735,05	54.735,05	54.735,05	0,00	731,29
u724_125	724	125	38.976,76	38.976,76	38.977,34	38.982,58	0,00	1.642,23
u724_200	724	200	28.079,97	28.079,97	28.084,06	28.094,74	0,01	1.634,34
rl1304_010	1304	10	2.147.312,37	2.147.312,37	2.178.140,10	2.205.079,52	1,44	306,74
rl1304_050	1304	50	802.283,41	802.283,41	807.158,48	810.831,95	0,61	1.847,65
rl1304_100	1304	100	498.090,74	498.090,74	498.379,82	499.563,12	0,06	2.847,27
rl1304_200	1304	200	276.977,60	276.977,60	276.983,18	277.033,42	0,00	2.176,40
rl1304_300	1304	300	191.224,85	191.224,85	191.235,17	191.276,99	0,01	2.324,17
pr2392_020	2392	20	2.238.410,94	2.238.410,94	2.257.901,36	2.283.143,48	0,87	642,41
pr2392_075	2392	75	1.092.294,02	1.092.294,02	1.098.782,83	1.110.326,37	0,59	1.642,55
pr2392_150	2392	150	711.111,25	711.111,25	711.451,32	712.032,07	0,05	3.249,95
pr2392_300	2392	300	458.145,29	458.145,29	458.174,94	458.211,10	0,01	4.680,39
pr2392_500	2392	500	316.042,97	316.042,97	316.080,44	316.146,21	0,01	7.994,38
fnl4461_0020	4461	20	1.285.523,48	1.285.523,48	1.293.124,80	1.302.484,38	0,59	3.508,01
fnl4461_0100	4461	100	550.043,01	550.043,01	551.465,45	552.800,36	0,26	7.504,90
fnl4461_0250	4461	250	336.109,51	336.109,51	336.422,54	337.288,29	0,09	8.059,13
fnl4461_0500	4461	500	224.671,57	224.672,12	224.703,07	224.788,18	0,01	8.184,83
fnl4461_1000	4461	1000	145.862,38	145.862,38	145.866,63	145.872,88	0,00	6.608,94

Segundo os dados reportados na Tabela 9 é possível observar que a heurística de obtenção de soluções iniciais (fase 1) é mais eficiente quanto maior for a relação *n*/*p*. Também é possível observar que a média mais elevada do *gap* ocorreu no conjunto de instâncias ali535 (também observados nas fases 2 e 3), o que se deve ao fato de que este conjunto de instâncias possui alguns pontos de demanda dispersos e outros muito agrupados em determinadas regiões, o que dificulta a resolução. A utilização de um número maior de iterações proporciona melhora na grande maioria das instâncias, mas a relação entre tempo computacional e melhora do valor de função objetivo nem sempre é eficiente.

Na fase 2 (Tabela 10), observa-se que a qualidade da solução é boa, pois na maioria das instâncias a solução encontrada teve um *gap* médio menor de 3%, com valor médio total do *gap* igual a 1,07%. Ainda, mesmo que em alguns casos a limitação de tempo na exploração da vizinhança não permita que se tenha garantia que no subconjunto analisado não existam movimentos de melhora, o tempo permitido para processamento é suficiente para que se tenha uma boa exploração da vizinhança.

Como as soluções obtidas na segunda fase já são relativamente satisfatórias, dependendo da necessidade de tomada de decisão, a terceira fase pode ser suprimida ou compreendida como uma fase de pós-otimização, neste caso, permitindo explorar mais a fundo a solução corrente ou determinadas regiões.

Os dados da Tabela 11 mostram que a utilização das reduções R1b+R3+R4 guiada pelo algoritmo iterativo (fase 3) com os parâmetros estabelecidos mostrou-se eficiente na capacidade de melhorar as soluções encontradas na fase 2 e, ainda, em obter as melhores soluções conhecidas para estas instâncias. Cabe observar que em geral quanto maior for a relação *n*/*p* há uma tendência do CPLEX provar a otimalidade do modelo matemático reduzido ou até mesmo o completo num tempo inferior. No entanto, os valores de função objetivo das soluções encontradas em todos os testes são mais dispersos, o que pode ser observado no *gap* médio. Embora a maioria das soluções reportadas na Tabela 11 podem ser melhoradas, estão mais propícias a grandes melhoras as soluções das instâncias com maior relação *n*/*p*.

Na Tabela 12 são apresentados alguns dados sobre o processo de redução de variáveis, onde é possível observar a quantidade de variáveis do modelo matemático eliminadas na segunda fase do processo de resolução com os parâmetros citados anteriormente. Nas três primeiras colunas da Tabela 12 temos o nome da instância, a quantidade de nós e medianas, respectivamente. A coluna VarTotal mostra o total de variáveis para a referida instância, que é dado por n^2 para o modelo matemático descrito na seção 3.2. A coluna VarElim mostra a

quantidade de variáveis eliminadas na fase 2 do processo de resolução com as reduções R1 e R4. A coluna VarRest exibe a quantidade de variáveis do modelo reduzido. Por fim, as duas últimas colunas trazem os tamanhos dos arquivos em MB no formato .lp exportados do CPLEX para o modelo reduzido e o modelo completo, respectivamente.

Para os dois maiores grupos de instâncias não foi possível exportar o modelo, devido as restrições de memória do computador.

Tabela 12 – Quantidade de variáveis eliminadas na segunda fase.

Instância	*n*	*p*	VarTotal	VarElim (%)	VarRest	lpRedu (MB)	lpFull (MB)
lin318_005	318	5	101.124	54,53	45.983	4,79	10,67
lin318_015	318	15	101.124	83,00	17.192	1,70	10,67
lin318_040	318	40	101.124	93,58	6.492	0,61	10,67
lin318_070	318	70	101.124	96,31	3.728	0,34	10,67
lin318_100	318	100	101.124	97,32	2.706	0,24	10,67
ali535_005	535	5	286.225	95,77	12.113	1,17	31,28
ali535_025	535	25	286.225	95,37	13.258	1,28	31,28
ali535_050	535	50	286.225	95,54	12.771	1,22	31,28
ali535_100	535	100	286.225	97,35	7.581	0,70	31,28
ali535_150	535	150	286.225	98,18	5.223	0,48	31,28
u724_010	724	10	524.176	96,86	16.474	1,64	59,37
u724_030	724	30	524.176	96,54	18.123	1,80	59,37
u724_075	724	75	524.176	96,59	17.877	1,76	59,37
u724_125	724	125	524.176	97,91	10.931	1,03	59,37
u724_200	724	200	524.176	98,67	6.978	0,65	59,37
rl1304_010	1304	10	1.700.416	98,26	29.550	3,02	200,29
rl1304_050	1304	50	1.700.416	98,16	31.267	3,16	200,29
rl1304_100	1304	100	1.700.416	98,18	30.914	3,07	200,29
rl1304_200	1304	200	1.700.416	98,71	21.965	2,12	200,29
rl1304_300	1304	300	1.700.416	99,12	14.992	1,41	200,29
pr2392_020	2392	20	5.721.664	98,96	59.610	6,27	–
pr2392_075	2392	75	5.721.664	98,95	59.982	6,24	–
pr2392_150	2392	150	5.721.664	98,96	59.307	6,12	–
pr2392_300	2392	300	5.721.664	99,14	49.460	5,03	–
pr2392_500	2392	500	5.721.664	99,47	30.327	2,98	–
fnl4461_0020	4461	20	19.900.521	99,42	115.676	12,28	–
fnl4461_0100	4461	100	19.900.521	99,44	111.443	11,77	–
fnl4461_0250	4461	250	19.900.521	99,44	111.872	11,76	–
fnl4461_0500	4461	500	19.900.521	99,48	102.510	10,61	–
fnl4461_1000	4461	1000	19.900.521	99,73	52.813	5,34	–

Outra observação relevante é a quantidade de variáveis do modelo reduzido. Embora dentro de um mesmo grupo haja significavas diferença nas quantidades de variáveis é possível

observar que a média de variáveis do modelo reduzido no grupo tem uma relação com o número de nós da instância. Isto pode ser observado no gráfico representado na Figura 9, no qual o eixo *x* representa a quantidade de nós e o eixo *y* a média de variáveis do modelo reduzido. A relação é aproximadamente linear para a configuração dos parâmetros estabelecidos, variando apenas no primeiro grupo de instâncias, pois esta não utiliza a redução R4, já que a quantidade de nós é relativamente pequena.

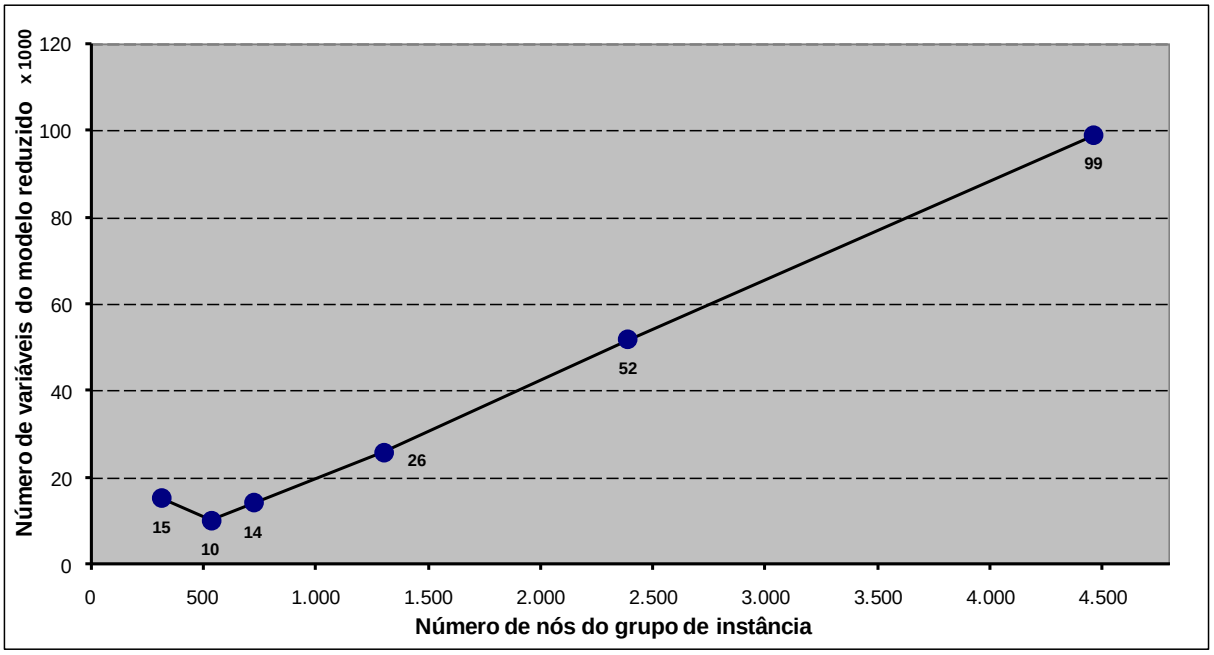

Figura 9 – Relação da média de variáveis restantes num grupo com o número de nós da instância.

Outra observação passível de ser reportada refere-se a comparação do desempenho de uma busca local com a vizinhança de Osman e Christofides (1994), com as técnicas de redução de variáveis propostas neste trabalho. Observou-se que, principalmente para as instâncias de grande porte, a busca local toma um tempo muito superior para a exploração da vizinhança do que as estratégias utilizadas neste trabalho, evidenciando assim o bom desempenho da técnica de redução de variáveis.

4 PROBLEMA DE PROGRAMAÇÃO DE TAREFAS EM MÁQUINAS PARALELAS

4.1 Introdução

O problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência considera um conjunto de *n* tarefas que devem ser, uma a uma, processadas sem interrupção por exatamente uma única máquina dentre *m* disponíveis. No caso em que as máquinas são não relacionadas, o tempo de processamento de uma tarefa depende da máquina na qual a mesma é processada. Os tempos de preparação de máquina (*setup time*) para este problema são dependentes da máquina e da sequência, isto é, o *setup time* computado na máquina *i* entre as tarefas *j* e *k* é diferente do *setup time* computado na mesma máquina entre as tarefas *k* e *j*. Ainda, o *setup time* entre as tarefas *j* e *k* na máquina *i* é diferente do *setup time* entre as tarefas *j* e *k* na máquina *i'*. O objetivo é encontrar uma distribuição das tarefas às máquinas de tal modo que o tempo de processamento total da máquina com maior carga entre todas (*makespan*) seja o menor possível. Esse problema é denotado por $R|S_{ijk}|C_{max}$ em Pinedo (2008), seguindo a classificação de três campos introduzida por Graham et al. (1979).

Máquinas não uniformes representam mais adequadamente problemas encontrados na prática e são generalizações dos problemas de programação em máquinas idênticas e relacionadas. Tempos de preparação de máquinas durante a produção envolve trabalho e tempo relativo à limpeza de ferramentas e do ambiente. O *setup time* para essas atividades é frequentemente negligenciado e incorporado no tempo de processamento, ou seja, ignorado, no entanto, em alguns casos, devido ao impacto na solução final, deve ser calculado separadamente do tempo de processamento das tarefas. Allahverdi et al. (1999) apresentam o estado da arte de sequenciamento de tarefas em máquinas com *setup time*. Em Yu et al. (2002), os autores afirmam que programação de tarefas em máquinas não relacionadas é um dos problema mais difíceis, com respeito à complexidade, considerando programação de tarefas em máquinas idênticas, relacionadas e não-relacionadas.

Existem vários trabalhos na literatura que abordam o problema de programação de tarefas em máquinas não relacionadas, como França et al. (1996), Weng et al. (2001), Kim et al. (2002), Chen (2005), Low (2005), Chen (2006), Chen e Wu (2006), Rabadi et al. (2006), De Paula et al. (2007), Logendran et al. (2007) e Armentano e De França (2007), para citar

alguns. Estes estudos consideram a otimização de um ou mais critérios de avaliação, como minimização do adiantamento e/ou atraso e *makespan*. Especificamente para o problema em questão, Vallada e Ruiz (2011) apresentam um algoritmo genético e propõem um conjunto de instâncias com características diversas.

Neste capítulo é proposta uma vizinhança de grande porte baseada em programação inteira mista (PIM) que é explorada por otimizador de programação matemática genérico. A intensidade da busca na vizinhança pode variar alterando o valor dos parâmetros de tempo e tamanho associados aos respectivos problemas de PIM. Resultados comparativos são apresentados com relação ao algoritmo genético proposto em Vallada e Ruiz (2011).

O restante do capítulo está organizado da seguinte forma. Na seção seguinte é apresentado o modelo matemático proposto por Vallada e Ruiz (2011) para o problema de programação de tarefas em máquinas não relacionadas com tempo de preparação dependente da sequência e da máquina com objetivo de minimizar o *makespan*. Na seção 4.3 são descritas duas heurísticas construtivas. Na seção 4.4 é apresentada uma vizinhança de grande porte baseada em PIM, uma estratégia de redução de variáveis do modelo matemático da vizinhança, uma estratégia de diversificação, além de outras considerações importantes. Na sequência, é definida uma busca local e um algoritmo de múltiplos inícios e, por fim, os resultados computacionais são discutidos.

4.2 Modelo matemático

Nesta seção é apresentado um modelo de programação inteira mista para o problema de programação de tarefas em máquinas não relacionadas com tempo dependente da sequência e da máquina com objetivo de minimizar o *makespan*, de acordo com Vallada e Ruiz (2011).

Para simplificar a exposição do modelo, faz-se necessário definir a notação utilizada.

Parâmetros

n : número de tarefas.

m : número de máquinas.

M : $\{1,...,m\}$, representa o conjunto das máquinas.

N : $\{1,...,n\}$, representa o conjunto das tarefas.

V : um número suficientemente grande.

p_{ij} : tempo de processamento da tarefa j, $j \in N$, na máquina i, $i \in M$.

s_{ijk} : tempo de preparação (*setup time*) da máquina i, $i \in M$, para execução da tarefa k após o processamento da tarefa j, k, $j \in N$.

Variáveis

$$x_{ijk} = \begin{cases} 1 & \text{se a tarefa } j \text{ precede a tarefa } k \text{ na máquina } i \\ 0 & \text{caso contrário} \end{cases}$$

$C_{ij} \geq 0$: Tempo de execução das tarefas j na máquina i, $i,j \in N$.

$C_{max} \geq 0$: Tempo máximo de execução.

$$\text{Min } C_{max} \tag{1}$$

Sujeito a:

$$\sum_{i \in M} \sum_{\substack{j \in \{0\} \cup \{N\} \\ j \neq k}} x_{ijk} = 1 \qquad \forall k \in N \tag{2}$$

$$\sum_{i \in M} \sum_{\substack{k \in N \\ j \neq k}} x_{ijk} \leq 1 \qquad \forall j \in N \tag{3}$$

$$\sum_{k \in N} x_{i0k} \leq 1 \qquad \forall i \in M \tag{4}$$

$$\sum_{\substack{h \in \{0\} \cup \{N\} \\ h \neq k, h \neq j}} x_{ihj} \geq x_{ijk} \qquad \forall j,k \in N,\ j \neq k, \forall i \in M \tag{5}$$

$$C_{ik} + V(1 - x_{ijk}) \geq C_{ij} + s_{ijk} + p_{ik} \qquad \forall j \in \{0\} \cup \{N\}, \forall k \in N,\ j \neq k, \forall i \in M \tag{6}$$

$$C_{i0} = 0 \qquad \forall i \in M \tag{7}$$

$$C_{ij} \geq 0 \qquad \forall j \in N, \forall i \in M \tag{8}$$

$$C_{\max} \geq C_{ij} \qquad \forall j \in N, \forall i \in M \tag{9}$$

$$X_{ijk} \in \{0,1\} \qquad \forall j \in \{0\} \cup \{N\}, \forall k \in N,\ j \neq k, \forall i \in M \tag{10}$$

O objetivo é minimizar o *makespan* (C_{max}). O conjunto de restrições (2) garante que cada tarefa seja atribuída a uma única máquina e tenha exatamente um antecessor. As restrições (3) determinam que cada tarefa só pode ter um sucessor. É importante observar que uma tarefa artificial 0 identifica o início do sequenciamento das tarefas nas máquinas, e o conjunto de restrições (4) limita o número de sucessores da tarefa artificial em, no máximo, um para cada máquina. O conjunto (5) garante que as tarefas são sequenciadas corretamente

nas máquinas. As restrições (6) determinam o tempo final do processamento das tarefas de cada máquina. Os conjuntos de restrições (7) e (8) definem o tempo de conclusão como 0 para as tarefas artificiais e valores não-negativos para as demais tarefas, respectivamente. As restrições (9) definem o tempo máximo de execução da máquina mais carregada ou *makespan*. Finalmente, o conjunto de restrições (10) define as variáveis binárias.

4.3 Heurísticas construtivas

Tanto para a busca local quanto para o algoritmo de múltiplos inícios descrito na seção 4.5, é necessário gerar uma solução inicial. Para a obtenção dessa solução inicial foram implementadas duas heurísticas construtivas. A primeira heurística gera uma solução denominada solução inicial trivial, e consiste em tomar as tarefas na ordem dos índices atribuídos a cada uma e inser-las sequencialmente nas máquinas, de forma que a tarefa com índice k seja designada para a máquina com índice (k mod m). Embora este método gere soluções de baixa qualidade, ele permite observar a robustez que o uso da vizinhança de grande porte propicia. Ou seja, se é possível encontrar soluções de boa qualidade mesmo partindo de soluções triviais.

No segundo método, as tarefas são designadas para as máquinas nas quais produzem menor acréscimo de carga, seguindo uma ordenação pré-estabelecida. Para a definição da ordem em que as tarefas são alocadas às máquinas é avaliada a soma de todos os custos de designação de cada tarefa para qualquer máquina, sendo esta a primeira tarefa ou sucessora de outra tarefa. A ordem que as tarefas são alocadas é a ordem decrescente de classificação. Deste modo, quando o aumento de *makespan* é inevitável, a tarefa é inserida na máquina que fornecer o menor acréscimo de *makespan*. A solução gerada por esse método foi denominada solução inicial gulosa. A ideia subjacente consiste em favorecer que as últimas tarefas a serem designadas tenham custos menores, de forma a aumentar as chances de inserção de uma tarefa em uma máquina sem que ocorra incremento no valor do makespan.

Duas variações dos métodos construtivos também são definidos: construtivo trivial aleatório e construtivo guloso aleatório. A primeira é uma variação do método construtivo de obtenção da solução inicial trivial e o segundo é uma variação do método de obtenção da solução inicial gulosa. A diferença destes aos respectivos métodos de origem é que a tarefa

escolhida para ser inserida na máquina é feita aleatoriamente dentre as tarefas que ainda não foram inseridas.

4.4 Vizinhança de grande porte baseada em programação inteira mista

Esta seção descreve uma vizinhança de grande porte baseada em programação inteira mista para definir um conjunto de sequências de movimentos atômicos denominados inserção e expulsão. O propósito reside em utilizar um modelo matemático para identificar conjuntos destas sequências de movimentos e obter uma melhora no valor da função objetivo.

No movimento atômico de inserção, indexado por uma variável de decisão y_{ij}, a tarefa *j* é inserida como sucessora da tarefa *i*. Já no movimento atômico de expulsão, representado pela variável de decisão x_{ij}, a tarefa *j* deve tomar o lugar da tarefa *i*, e esta última pertencerá a outro movimento que poderá ser uma inserção ou uma nova expulsão.

A vizinhança é definida por dois tipos de sequência de movimentos denominados cadeia e ciclo, representados na Figura 10 a partir de uma solução do problema em estudo com duas máquinas, ver (a). Um movimento expulsão é ilustrado em (b) e uma sequência ciclo, definida apenas por movimentos deste tipo, pode ser observada envolvendo as primeiras tarefas de cada máquina em (d). Em uma sequência de movimentos em cadeia, após a expulsão da primeira tarefa, esta pode ser tanto inserida como sucessora de outra tarefa (c) quanto expulsar outra tarefa e assim sucessivamente até que a sequência finalize com uma inserção, como pode ser observado em (d). É importante entender que a vizinhança considera uma solução estática, e, para que o custo de cada movimento atômico seja válido, algumas tarefas não podem ser movimentadas, aquelas assinaladas com um “x” em (d). Estas tarefas são sucessoras ou antecessoras das tarefas que estão sendo movimentadas.

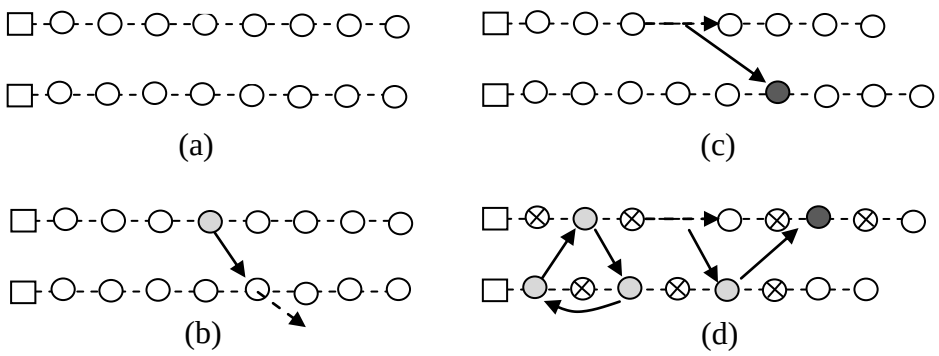

Figura 10 – Ilustração de movimentos de troca, inserção, ciclo e cadeia.

Um grande número de sequências independentes de movimentos pode ser definido, incluindo movimentos entre tarefas de uma mesma máquina e entre máquinas diferentes.

4.4.1 Modelo matemático

Formalmente, o modelo matemático associado à vizinhança é definido a seguir:

Parâmetros:

N^* : $\{1,...,n,n+1,...,n+m\}$, representa o conjunto das tarefas a serem executadas, bem como m tarefas artificiais que representam cada máquina.

D : $N^* \setminus N = \{n+1,...,n+m\}$, representa o conjunto de tarefas artificiais.

U_m : Conjunto das tarefas da solução inicial designadas à máquina m, $m \in M$.

cO_i : Custo da retirada da tarefa i de sua posição.

cX_{ij} : Custo da inserção da tarefa j na posição da tarefa i.

cY_{ij} : Custo da inserção da tarefa j como sucessora da tarefa i.

cW_i : Custo de definir o sucessor da tarefa i como sucessor da tarefa que precede i.

cM_m : Custo total da máquina m da solução corrente, com $m \in M$.

$suc(i)$: Representa a tarefa sucessora da tarefa i no sequenciamento de uma das máquinas.

Variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se a tarefa } j \text{ substitui e expulsa a terefa } i \text{ de sua posição.} \\ 0 & \text{caso contrário} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{se a tarefa } j \text{ é inserida como sucessora da tarefa } i \\ 0 & \text{caso contrário} \end{cases}$$

$$w_i = \begin{cases} 1 & \text{o sucessor da tarefa } j \text{ passa a suceder o predecessor da tarefa } j. \\ 0 & \text{caso contrário} \end{cases}$$

$C_{max} \geq 0$: Tempo máximo de execução da máquina mais carregada (*makespan*).

$$\text{Min } C_{max} \qquad (1)$$

Sujeito a:

$$\sum_{j \in N} x_{ij} + \sum_{j \in N} y_{ij} + w_i \leq 1 \qquad \forall i \in N \qquad (2)$$

$$\sum_{j \in N} y_{ij} \leq 1 \qquad \forall i \in D \qquad (3)$$

$$\sum_{i \in N} x_{ij} + \sum_{i \in N^*} y_{ij} \leq 1 \qquad \forall j \in N \qquad (4)$$

$$\sum_{\substack{j \in N \\ i \notin D}} x_{ij} + \sum_{j \in N} y_{ij} + \sum_{j \in N} x_{j,suc(i)} + \sum_{j \in N^*} y_{j,suc(i)} \leq 1 \qquad \forall i \in N^* \qquad (5)$$

$$\sum_{i \in N} x_{ij} + \sum_{i \in N^*} y_{ij} + \sum_{i \in N} x_{i,suc(j)} + \sum_{i \in N^*} y_{i,suc(j)} \leq 1 \qquad \forall j \in N \qquad (6)$$

$$\sum_{j \in N} x_{ij} \leq \sum_{j \in N} x_{ji} + \sum_{j \in N^*} y_{ji} \qquad \forall i \in N \qquad (7)$$

$$\sum_{j \in N^*} y_{ji} + \sum_{j \in N} x_{ji} \leq \sum_{j \in N} x_{ij} + w_i \qquad \forall i \in N \qquad (8)$$

$$\sum_{j \in N} y_{ij} + \sum_{j \in N} x_{ji} + \sum_{j \in N^*} y_{ji} \leq 1 \qquad \forall i \in N \qquad (9)$$

$$\sum_{\substack{i \in N \\ j \leq n}} cO_j x_{ij} \sum_{\substack{i \in N \\ j \leq n}} cX_{ji} x_{ji} + \sum_{\substack{i \in N^* \\ j \neq m+n}} cO_j y_{ij} +$$

$$+ \sum_{i \in N} cY_{ji} y_{ji} + \sum_{i \in U_m} cW_i w_i + cM_m \leq C_{\max} \qquad \forall j \in U_m,\ \forall m \in M \qquad (10)$$

$$\sum_{j \in N} x_{ji} + \sum_{j \in N} y_{ji} - w_i \geq 0 \qquad \forall i \in N \qquad (11)$$

$$C_{max} \geq 0,\ x_{ij}, y_{kj}, w_i \in \{0,1\} \qquad \forall i, j \in N, \forall k \in N^* \qquad (12)$$

O objetivo é obter movimentos que minimizem o tempo máximo de execução ou *makespan*. As restrições (2), (3) e (4) obrigam que uma tarefa participe apenas de um

movimento. Os conjuntos de restrições (5) e (6) definem que se uma tarefa for movimentada, as tarefas sucessoras e antecessoras devem se manter fixas. Esta restrição é necessária para que os custos de inserção e expulsão de uma tarefa sejam calculados corretamente, visto que os mesmos são dados em função das tarefas que precedem e sucedem a tarefa. Em (7) e (8) são definidas as restrições que forçam uma tarefa que compõem um movimento ser alocada em outra posição. Especificamente sobre as restrições (8), se uma tarefa for retirada da sua posição original uma outra tarefa deve ocupar essa posição ou o antecessor e o sucessor devem ser conectados na sequência. Este fato é representado pelas variáveis denotadas por *x* e *w* no lado direito da desigualdade. As restrições (9) implicam que, se uma tarefa for inserida como sucessora de outra, a tarefa sucessora da tarefa inserida deve permanecer fixa, novamente, assim como nas restrições (5) e (6), isto é necessário para que o custo de um movimento atômico seja avaliado corretamente. De outra forma, também implica que se uma tarefa for movimentada, nenhuma tarefa pode ser inserida como sua predecessora. As restrições (10) são responsáveis por avaliar os custos dos movimentos. As restrições (11), a rigor, podem ser suprimidas, uma vez que tem como função evitar que variáveis *w* assumam valor 1 sem que a correspondente tarefa faça parte de algum movimento. De fato, caso sejam suprimidas, o único inconveniente que decorre é a necessidade de analisar quais variáveis *w* com valor 1 não compõe a solução do modelo. Por último, no conjunto de restrições (12), tem-se a restrição que impõe as condições para o domínio das variáveis do problema.

Os parâmetros que representam os custos de retirada e inserção de tarefas são definidos a partir da solução corrente, em que *pre*(*i*) representa a tarefa que precede a tarefa *i* e $Q_{i,j} = p_{ij} + s_{kij}$ representa o custo total de alocação da tarefa *j* como sucessora da tarefa *i* na máquina *k*. Assim, $cO_j = -Q_{pre(j)j} - Q_{j,suc(j)}$ é o custo da retirada da tarefa *j* de sua posição na solução corrente. Esta situação ocorre tanto com as variáveis x_{ij} quando a tarefa *i* é expulsa, quanto nas variáveis y_{ij} quando a tarefa *j* é retirada de sua posição. Já o custo denotado por $cX_{ij} = Q_{pre(i),j} + Q_{j,suc(i)}$ é aplicável apenas nas variáveis x_{ij}, pois representa o custo de inserção da tarefa *j* na posição da tarefa *i*. O custo da inserção de uma tarefa *j* como sucessora de uma tarefa *i* é dado por $cY_{ij} = -Q_{i,suc(i)} + Q_{i,j} + Q_{j,suc(i)}$, e é associado exclusivamente as variáveis y_{ij}. Por fim, o custo $cW_i = Q_{pre(i),suc(i)}$ é o custo de associar a tarefa antecessora com a sucessora de uma tarefa *i*, e é associado às variáveis w_i.

As variáveis denotadas por *w* auxiliam na definição de uma sequência de movimentos em cadeia, na qual a primeira tarefa expulsa é indexada pela variável w_i. Além disso, esta

variável indica que o arco entre a tarefa predecessora e sucessora da tarefa *i* deve ser fechado, ou seja, a tarefa sucessora da tarefa *i* deve passar a ser sucessora da tarefa que precede a tarefa *i*.

4.4.2 Método de redução de variáveis

Assim como no PPMC, para o qual foram propostos métodos de redução de variáveis na seção 3.3, também é proposta uma redução de variáveis para o modelo matemático da vizinhança de grande porte definido para o problema de programação de tarefas em máquinas paralelas. Uma boa estratégia de redução proporciona uma melhora no desempenho do otimizador, de forma que o tempo para a resolução do problema tende reduzir.

A ideia desta redução é eliminar variáveis que representam movimentos que têm pequena probabilidade de pertencerem à solução ótima. Para tal, são definidos ω_y e ω_x dados pela fórmula a seguir:

$$\omega_y = \Omega * \left(\sum_{\substack{i \in N^*, j \in N, \\ i \neq j, j \neq suc(i)}} qy_{i,j} \right) / n_y, \quad \omega_x = \Omega * \left(\sum_{\substack{i,j \in N \\ i \neq j, j \neq suc(i)}} qx_{i,j} \right) / n_x,$$

em que n_y e n_x são respectivamente o número de movimentos atômicos de inserção e expulsão possíveis. O parâmetro Ω representa um fator de expansão ou redução da média dos custos dos movimentos atômicos, o qual serve para definir a quantidade de variáveis a ser eliminada. Apesar da possibilidade de definir diferentes valores do parâmetro Ω para os diferentes movimentos atômicos, optou-se por definí-lo de forma única. Os custos de movimentação de tarefas são definidos por $qy_{i,j} = cO_j + cY_{i,j} + cW_j$ e $qx_{i,j} = cX_{ij}$.

Do exposto, os parâmetros ω_y e ω_x representam um limitante superior na avaliação dos custos, ou seja, dada uma variável x_{ij}, se o custo $qx_{ij} > \omega_x$, então a variável é simplesmente eliminada. Analogamente, uma variável y_{ij} é descartada se $qy_{ij} > \omega_y$, e passa a não figurar no modelo matemático.

4.4.3 Método de diversificação

Mesmo que a vizinhança de grande porte facilite a busca por boas soluções em diferentes bacias atratoras do domínio, há situação em que determinado vales são difíceis de serem superados, de forma que é necessária a utilização de algum método capaz de proporcionar a exploração de novas áreas em busca de soluções de melhor qualidade.

Uma estratégia encontrada para fugir de ótimos locais utilizando a vizinhança proposta, corresponde a obrigar que sejam executados um determinado número de movimentos mesmo que haja piora da solução corrente. Assim, quando é desejável que seja executada a diversificação a restrição (13) representada a seguir é adicionada ao modelo.

$$\sum_{j \in N} x_{ji} + \sum_{j \in N^*} y_{ji} \geq \lambda \qquad \forall i \in N \tag{13}$$

O parâmetro λ define o número mínimo de movimentos atômicos a serem executados, e por padrão foi utilizado um número aleatório entre 10% e 40% do número de tarefas da instância, o que ajuda a evitar que ocorram ciclos durante as iterações de diversificação. Além disso, é adicionado ao lado direito das restrições (10) uma constante Φ que limita o quanto é possível piorar a solução corrente. Esta constante pode ser definida como um número grande, pois assim permite que o otimizador encontre facilmente uma solução factível em tempo reduzido, e a partir daí encontre novas soluções com melhor avaliação. Se for definido um valor baixo para esse parâmetro, corre-se o risco de que não sejam encontradas soluções factíveis mesmo com elevados tempos de processamento, o que não é desejável. Apesar de Φ ser mais um parâmetro que necessita de configuração, o ajuste é relativamente fácil de ser definido, pois um valor elevado não influência significativamente na resolução do modelo. Com isto em mente, definiu-se Φ de modo a permitir uma piora de até 100% da solução corrente, ou seja, a constante somada é o valor do *makespan* da solução corrente.

4.4.4 Outras considerações

O número de variáveis do modelo matemático utilizado para explorar a vizinhança é dependente principalmente do número de tarefas. Para as variáveis *y*, o número de variáveis é dado pela expressão $n^2 + n(m-2)$, onde n e m representam respectivamente o número de tarefas e máquinas da instância. Para as variáveis *x*, o número de variáveis é $n^2 - 3n + 2m$, e

para as variáveis *w*, sua quantidade é exatamente *n*. Assim, levando em conta ainda a variável C_{max}, a quantidade total de variáveis é dada por $2n^2 + n(m-4) + 2m + 1$.

A Figura 11 mostra um gráfico de barras verticais com as quantidades de variáveis em milhares de unidade, para diferentes composições de tarefas e máquinas representadas no eixo horizontal. A altura do primeiro segmento da barra representa a quantidade de variáveis *y*, o segundo segmento representa a quantidade de variáveis *x* e, no topo da barra, as variáveis *w*, que devido a pequena quantidade ficam praticamente imperceptíveis diante da grande quantidade das variáveis *x* e *y*. Assim, com a soma dos três segmentos temos a quantidade total de variáveis para cada composição.

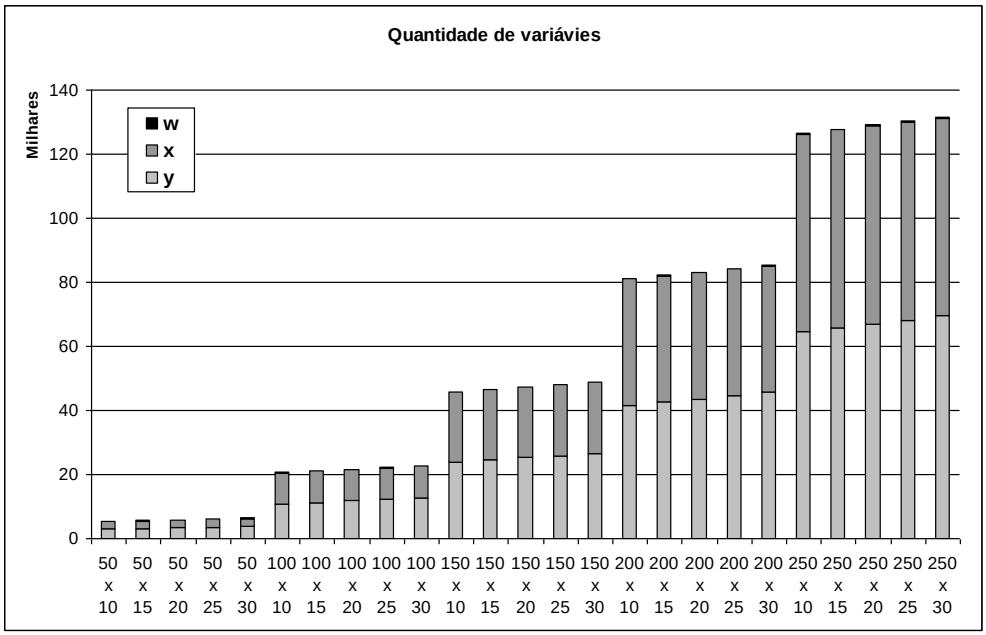

Figura 11 – Número de variáveis para diferentes quantidades de tarefas e máquinas.

Uma consideração importante é o tempo computacional destinado a resolução do modelo matemático através do otimizador. Em geral, para instâncias acima de 30 tarefas o otimizador tem dificuldade de provar a otimalidade da solução do modelo matemático. Muito desta dificuldade é imposta pelas restrições (10), pois os coeficientes destas restrições são não unitários e, além disso, estas restrições são responsáveis por avaliar os custos das sequências de movimentos. Esta dificuldade impõe limitações à resolução completa do modelo, necessitando assim de parâmetros que limitem o tempo de execução do otimizador para que o

tempo de resolução se mantenha dentro de um tempo aceitável. Apesar disso, em geral, o otimizador consegue encontrar sequências de movimentos em poucos segundos de processamento, o que é altamente desejável.

O modelo matemático da vizinhança permite que sejam movimentadas, no máximo, pouco mais de 50% das tarefas, uma vez que o valor exato depende da quantidade de tarefas de cada máquina da solução corrente. Isso evidencia a grande diversidade de soluções possíveis de serem avaliadas e obtidas.

4.5 Busca local e Algoritmo de Múltiplos Inícios

A vizinhança proposta foi originalmente projetada para compor um procedimento de intensificação, diversificação ou pós-otimização de um método metaheurístico, visto que, principalmente em instâncias de grande porte, a resolução do modelo matemático associado pode requerer um tempo maior de processamento. Assim, a fim de avaliar a capacidade de obtenção de bons resultados, independentemente de uma avaliação mais cuidadosa com respeito ao tempo computacional, a vizinhança foi utilizada internamente a uma busca local. Desde modo, esta seção descreve uma busca local com e sem a estratégia de diversificação e um algoritmo de múltiplos inícios que utilizam a vizinhança de grande porte descrita na seção 4.4.

4.5.1 Busca local

Um algoritmo de busca local parte de uma solução obtida através de outro método, como um método construtivo, e iterativamente move-se para uma solução vizinha melhor que a corrente, até que um ótimo local seja encontrado.

Especificamente para o caso em estudo, a busca na vizinhança é resolvida através do otimizador, retornando uma sequência de movimentos atômicos que após executados geram nova solução. O processo é repetido até que mais nenhuma sequência de movimentos seja identificada pela vizinhança ou os parâmetros sejam excedidos.

Dois parâmetros são definidos para a busca local: o primeiro é o tempo máximo (*tmax*) que o otimizador permanece na busca dos movimentos; o segundo é o tempo mínimo (*tmin*).

O tempo máximo é necessário principalmente para as instâncias de grande porte, uma vez que o otimizador pode ter dificuldade de provar a otimalidade num tempo viável. Este procedimento é adotado para manter o tempo computacional dentro de um limite aceitável, no entanto, caso seja necessário tentar soluções de melhor qualidade, nada impede que mais tempo seja utilizado. Partindo da solução inicial, a vizinhança é iterada até que seja provada a otimalidade da vizinhança sem melhora de função objetivo ou o tempo máximo seja excedido.

O segundo parâmetro, *tmin*, determina o tempo mínimo que o otimizador deve ficar buscando por melhores soluções, no caso em que a otimalidade não tenha sido provada. Após este tempo, caso o otimizador tenha conseguido encontrar movimentos que melhorem a solução inicial, a busca é interrompida, os movimentos são executados de forma a obter a nova solução, partindo para uma nova iteração da busca local. Existe a possibilidade de interromper a busca logo que a primeira solução é encontrada, no entanto, verificou-se que um pequeno tempo adicional em geral possibilita a obtenção de melhoras significativas que justificam sua utilização.

A Figura 12 ilustra o algoritmo de busca local proposto. Enquanto for possível atualizar a solução incumbente *S**, os procedimentos de redução de variáveis da vizinhança e busca na vizinhança são iterados (linhas 3 e 4, respectivamente). Ao final, a solução *S* é retornada (linha 6) com a melhor avaliação da função objetivo *f*(*S*).

```
algoritmo BUSCA_LOCAL(S, tmax, tmin)
1.faça
2.      S*←S;
3.      ReducaoVariaveis(S);
4.      BuscaVizinhanca(S, tmax, tmin);
5.enquanto(f(S)<f(S*))
6.retorna S;
fim BUSCA_LOCAL.
```

Figura 12 – Algoritmo de busca local.

Nota-se que a solução retornada não necessariamente representa um ótimo local. Esta condição é perdida quando uma busca pela vizinhança (linha 4) é interrompida pelo parâmetro *tmax*, ocasionando assim a perda da garantia da otimalidade na exploração da vizinhança e por consequência da busca local.

4.5.2 Busca local + diversificação

Foi observado que o desempenho da busca local pode ser melhor com o uso da estratégia de diversificação. Deste modo, nesta seção é proposto um algoritmo que itera procedimentos de busca local e diversificação com um máximo de IT_{max} iterações.

A Figura 13 ilustra o procedimento utilizando a estratégia de diversificação. A resolução parte de um possível ótimo local (linha 1) seguido da atualização da solução incumbente *S** (linha 2). Em cada iteração, até que o critério de parada é satisfeito, é realizada a resolução do modelo matemático com o método de diversificação, descrito na seção 4.4.3 (linha 4), com os parâmetros de tempo máximo, tempo mínimo, Φ e λ específicos do método de diversificação representados por um vetor de parâmetros *v*[ ]. Como esta estratégia tende a piorar a solução atual, a busca local é executada novamente (linha 5) e a solução incumbente é atualizada caso o valor da função objetivo da solução encontrada seja inferior à incumbente, já que o problema é de minimização. Por fim, a melhor solução é retornada.

```
algoritmo BUSCA_LOCAL_DIV(S, tmax, tmin, IT_max, v[ ])
1. BUSCA_LOCAL(S, tmax, tmin);
2. S* ← S;
3. para (Iter = 1, ..., IT_max) faça
4       Diversificacao(S, v[ ]);
5.      BUSCA_LOCAL(S, tmax, tmin);
6.      se( f(S) < f(S*)) então
7.          S* ← S;
8.      fim - se
9. fim - para
10. retorna S*;
fim BUSCA_LOCAL_DIV.
```

Figura 13 – Algoritmo de busca local + diversificação.

Esta estratégia faz uso da estrutura de diversificação para tentar fugir de ótimos locais. Em sua essência, baseia-se na metaheurística busca tabu, embora não considere qualquer estrutura de memória. Isto sugere que esta estrutura pode ser melhorada de modo a evitar ciclos, ainda que estes não sejam frequentes devido à complexidade e assimetria, por assim dizer, da sequência de movimentos usualmente aplicada à solução corrente.

4.5.3 Algoritmo de múltiplos inícios

Para instâncias de pequeno porte a busca local nem sempre encontra a melhor solução conhecida ou a solução ótima numa única execução, motivo pelo qual foi implementado um algoritmo de múltiplos inícios. Alia-se a este fato o tempo computacional relativamente baixo da exploração da vizinhança para este caso.

De maneira geral, o algoritmo de múltiplos inícios parte de diferentes soluções iniciais que são utilizadas como base para a execução da busca local. As soluções iniciais fornecidas ao algoritmo são, além da solução inicial gulosa, uma variação de (K_{iter} * n) / 2 soluções obtidas com o algoritmo construtivo guloso aleatório e (K_{iter} * n) / 2 soluções iniciais obtidas com o método construtivo trivial aleatório. Define-se K_{iter} como um parâmetro que determina o número de iterações do algoritmo de múltiplos inícios.

A Figura 14 traz em pseudocódigo o algoritmo de múltiplos inícios, no qual na primeira fase (linha 2 e 3) uma solução inicial é gerada pelo método construtivo guloso e na sequência a busca local é aplicada. A segunda e terceira fase são similares (linhas 4-10 e 11-17, respectivamente) e se diferenciam apenas pelo método de obtenção da solução inicial que é a construtiva gulosa aleatória e a construtiva trivial aleatória, respectivamente, seguidos da busca local e a atualização da solução incumbente. Por fim, é retornada a melhor solução encontrada dentre todas as K_{iter} * n + 1 iterações.

```
algoritmo MULTIPLOS_INICIOS(K_iter)
1. f(S*) ← ∞;
2. Const_Guloso(S);
3. BUSCA_LOCAL(S,tmax,tmin);
4. para (Iter = 1,...,K_iter * n/2) faça
5.      Const_Guloso_Aleatorio(S);
6.      BUSCA_LOCAL(S,tmax,tmin);
7.      se( f(S*) < f(S)) então
8.          S* ← S;
9.      fim - se
10. fim - para
11. para (Iter = K_iter * n/2 + 1,...,K_iter * n) faça
12.     Const_Trivial_Aleatorio(S);
13.     BUSCA_LOCAL(S,tmax,tmin);
14.     se( f(S*) < f(S)) então
15.         S* ← S;
16.     fim - se
17. fim - para
18. retorna S*;
fim MULTIPLOS_INICIOS.
```

Figura 14 – Algoritmo de múltiplos inícios.

Observa-se que para K_{iter} = 0 os passos nas linhas 4 a 19 não são executados, referindo-se assim a busca local na sua forma original, partindo da solução inicial gulosa.

Este algoritmo assemelha-se fortemente a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*), diferenciando-se unicamente no fato da heurística construtiva utilizar escolhas totalmente aleatórias, em vez de usar uma lista restrita de candidatos, gerando assim uma maior diversidade de soluções iniciais, o que é desejável já que são executas poucas iterações com, possivelmente, grande qualidade.

4.6 Resultados computacionais

Para os testes computacionais foi utilizado um computador com processador Intel Quad-Core Xeon X3360 2.83 GHz e o otimizador CPLEX 12.1 com sua configuração padrão. O conjunto de instâncias teste foi proposto em Vallada e Ruiz (2011), disponível em http://soa.iti.es, e é dividido em dois grupos: 640 instâncias consideradas de pequeno porte, 1000 instâncias consideradas de grande porte. Cada um dos grupos é subdividido em 10 instâncias com diferentes quantidades de máquinas e tarefas, além de diferentes intervalos para os tempos de *setup*.

Diferentes testes foram realizados para avaliar a vizinhança proposta. Primeiro, para as instâncias de pequeno porte, foi utilizado o algoritmo de múltiplos inícios. Como esta abordagem propiciou de imediato bons resultados, os esforços seguintes foram concentrados em testes com as instâncias de grande porte a partir da busca local, busca local + diversificação e redução de variáveis.

Para os testes, a não ser quando especificado, foi adotado $tmax = (n - m / 2) * 0{,}5$ e $tmin = 0{,}01 * tmax$, que demonstraram uma boa relação entre qualidade de solução e tempo computacional.

Para avaliação da qualidade das soluções obtidas foi definido o desvio médio relativo (*DMR*), de acordo com a fórmula $DMR = (S_h - S_g) / \min(S_h, S_g) * 100$, em que S_h corresponde à solução obtida através da busca local ou algoritmo de múltiplos inícios e S_g corresponde à melhor solução do algoritmo genético de Vallada e Ruiz (2011).

Para o algoritmo genético proposto em Vallada e Ruiz (2011) o critério de parada é o tempo computacional definido por $n * (m / 2) * t$, com um valor máximo para $t = 50$ milissegundos. Desta forma, o tempo de CPU máximo para a execução do algoritmo genético varia de 12,5 a 187,5 segundos em um computador com processador Intel Core 2 Duo, 2.4 GHz com 2 GB de memória RAM.

4.6.1 Resultados para as instâncias de pequeno porte

Os resultados do algoritmo de múltipos na resolução das instâncias de pequeno porte foram tabulados para diferentes valores do parâmetro K_{iter}, conforme ilustrado na Tabela 13. Entende-se que para $K_{iter} = 0$ tem-se a busca local partindo da solução inicial gulosa.

Na Tabela 13, a primeira coluna mostra o valor do parâmetro *k*. Os valores da coluna Tarefas representam um conjunto de 160 instâncias que possuem o mesmo número de tarefas e diferentes quantidades de máquinas, cujo *DMR* médio para este conjunto é mostrado na terceira coluna. A coluna seguinte apresenta o percentual de instâncias de cada grupo para as quais o algoritmo de múltiplos inícios conseguiu encontrar a melhor solução conhecida e, por fim, a coluna Tempo(s) traz o tempo computacional médio em segundos.

Tabela 13 – Resultados médios para instâncias de pequeno porte.

K_{iter}	Tarefas	*DMR*	Melhor Sol (%).	Tempo(s)
0	6	2,10	79,38	0,016
	8	5,22	57,50	0,048
	10	3,81	47,50	0,088
	12	4,01	34,38	0,187
Média		**3,79**	**54,69**	**0,085**
0,5	6	0,09	99,38	0,131
	8	0,12	98,13	0,435
	10	0,16	93,13	0,997
	12	0,22	88,75	2,413
Média		**0,15**	**94,84**	**0,994**
1	6	0,02	99,38	0,223
	8	0,00	100,00	0,772
	10	0,03	98,13	1,833
	12	0,03	96,88	4,553
Média		**0,02**	**98,59**	**1,845**
2	6	0,00	100,00	0,405
	8	0,00	100,00	1,428
	10	0,00	100,00	3,468
	12	-0,04	100,00	8,908
Média		**-0,01**	**100,00**	**3,552**

A partir dos dados da Tabela 13, é possível observar que a utilização da busca local (K_{iter} = 0), para as instâncias pequenas, na média, possibilitou obter boas soluções. Além disso, em 54,69% do total das instâncias foi possível obter o valor das melhores soluções conhecidas.

Fazendo uso do algoritmo de múltiplos inícios é possível perceber que com um pequeno acréscimo de esforço e tempo computacional obtém-se uma boa melhora nos valores do *DMR*, e do percentual das melhores soluções conhecidas. Além disso, para K_{iter} = 2 foi possível encontrar as melhores soluções conhecidas para todas as instâncias analisadas.

O tempo computacional máximo destinado ao algoritmo genético é 0,75, 1,00, 1,25 e 1,50 para o conjunto com 6, 8, 10 e 12 máquinas, respectivamente. Observa-se que quanto

maior a quantidade de tarefas e do fator K_{iter} a média de tempo utilizada pelo algoritmo de múltiplos inícios é maior se comparado ao utilizado pelo algoritmo genético. No entanto, o algoritmo de múltiplos inícios mostrou-se frequente em obter as melhores soluções conhecidas. Além disso, o foco principal foi os experimentos realizados para as instâncias de grande porte.

Outro fato a ser considerado é que na utilização do algoritmo de múltiplos inícios com $K_{iter} \geq 0{,}5$ foi possível observar a melhora no valor da função objetivo na terceira instância do grupo com 12 tarefas, 5 máquinas e *setup time* de 1 a 99, reportado como ótimo global por Vallada e Ruiz (2011). Isto explica o *DMR* negativo para último conjunto de instâncias de pequeno porte. Credita-se a melhora na solução da instância reportada aos valores de tolerância e precisão do CPLEX, além dos diferentes *hardwares* utilizados ou ainda por um erro de grafia.

Também foram testados valores maiores para K_{iter} do que os representados na Tabela 13, porém sem qualquer perspectiva de melhora, mostrando que as melhores soluções conhecidas tendem a serem as soluções ótimas.

4.6.2 Resultados para as instâncias de grande porte

Para as instâncias de grande porte foram aplicados diversos testes com diferentes estruturas de resolução e parâmetros. Alguns resultados dos testes computacionais como a utilização da busca local, busca local com estrutura de redução de variáveis e busca local + diversificação são aqui reportados.

Resultados da busca local

O primeiro teste realizado para o conjunto de instâncias de grande porte considera a busca local utilizando a vizinhança de grande porte. A Tabela 14 apresenta os resultados médios obtidos a cada conjunto de 40 instâncias, em que a primeira e a segunda coluna correspondem respectivamente ao número de tarefas e máquinas do grupo de instâncias. A coluna *DMR* traz o desvio médio relativo referente ao conjunto de cada configuração. A coluna MelhorSol traz o percentual de instâncias do conjunto para o qual a busca local

conseguiu igualar ou melhorar o valor da solução final encontrada pelo algoritmo genético proposto por Vallada e Ruiz (2011), e as colunas seguintes reportam o tempo médio em segundos para resolução de uma instância com o algoritmo genético e utilizando o método atual, respectivamente.

Tabela 14 – Resultados médios da busca local.

Tarefas	Máquinas	*DMR*	Melhor Sol (%)	Tempo-AG (s)	Tempo-Viz (s)
50	10	6,07	15,00	13	35,45
	15	-0,97	60,00	19	18,52
	20	-3,03	77,50	25	7,54
	25	-5,32	95,00	31	3,44
	30	-7,20	92,50	38	2,13
100	10	3,00	37,50	25	98,45
	15	-1,11	70,00	38	97,07
	20	-4,64	87,50	50	93,40
	25	-10,50	97,50	63	84,23
	30	-14,68	100,00	75	70,86
150	10	-1,82	77,50	38	177,43
	15	-3,34	87,50	56	164,00
	20	-6,66	95,00	75	158,52
	25	-12,71	100,00	94	152,47
	30	-13,76	100,00	113	137,99
200	10	-3,84	97,50	50	277,26
	15	-5,57	92,50	75	270,23
	20	-9,58	100,00	100	246,30
	25	-8,92	97,50	125	220,77
	30	-15,47	100,00	150	208,27
250	10	-5,47	97,50	63	386,37
	15	-8,24	100,00	94	369,51
	20	-2,11	92,50	125	379,14
	25	-3,54	90,00	156	336,23
	30	-3,65	87,50	188	314,58
Média		**-5,72**	**85,90**	**75**	**172,41**

Os resultados mostram que a busca local apresenta bom desempenho, pois conseguiu melhorar ou igualar 85,90% das melhores soluções conhecidas até então, sendo 79,6% estritamente melhores. O *DMR* médio das soluções obtidas para todos os conjuntos de instâncias com os parâmetros citados anteriormente foi de −5,72, e inclusive com reduções de até −33,33% para uma instância com 250 tarefas e 30 máquinas.

Durante os testes, como também na avaliação dos resultados, foi possível observar que quando a relação *n*/*m* é menor a busca local tem desempenho melhor, tanto em tempo computacional como em valor de função objetivo. Isso pode ser visualizado a partir dos dados reportados na Tabela 14 para as diferentes configurações de número de tarefas e máquinas, observando-se que a busca local apresentou os piores resultados para os conjuntos de instâncias 50x10 e 100x10.

Na maioria dos casos em que a busca local parte de uma solução de baixa ou média qualidade o CPLEX consegue encontrar movimentos que melhoram a função objetivo em um curto espaço de tempo, no entanto, há situações que isso não acontece. Em especial, isto ocorre na primeira iteração, na qual o CPLEX pode extrapolar o tempo limite máximo sem conseguir melhorar a solução inicial. Nos testes realizados com os parâmetros anteriores, das 1000 instâncias testadas, houve 13 destas situações, sendo uma no conjunto 200x25, três no conjunto 250x20, quatro no conjunto 250x25 e cinco no conjunto 250x30. Isso causou um impacto nos valores médios do *DMR*, pois, retirando esses casos, o *DMR* do conjunto restante de instâncias fica reduzido a −7,20%. Nestas instâncias, o fato de não conseguir melhorar a solução inicial num tempo inferior ao parâmetro de tempo máximo não implica que a vizinhança não consiga melhorar a solução ou que a solução inicial seja um ótimo local, pois, em testes adicionais, permitindo um tempo extra nestas instâncias, em todos os casos foi possível melhorar a solução inicial e inclusive melhorar a melhor solução conhecida, o que reduz o *DMR* para −7,30%.

Em outro experimento realizado foi verificado, como esperado, que quando a busca local parte da solução inicial trivial o otimizador despende mais tempo para encontrar as sequências de movimentos dentro dos parâmetros estabelecidos. No entanto, considerando somente as instâncias para as quais as sequências de movimentos são encontradas dentro do tempo máximo permitido, a média total do *DMR* fica em −5,69%, demonstrando que a capacidade da vizinhança não é intrinsecamente atrelada à solução inicial.

Outra observação é que, principalmente nas instâncias de grande porte, as quais o otimizador tem dificuldade de provar a otimalidade do modelo matemático associado à vizinhança, invariavelmente, somente a última iteração da busca local é limitada pelo tempo máximo de execução (*tmax*). Isto implica que a solução final de cada instância de larga escala é encontrada a, aproximadamente, 60% do tempo total utilizado. Este comportamento sugere a possibilidade de testar formas adaptáveis de atribuição do valor para o parâmetro *tmax*.

Resultados da busca local com redução

O teste seguinte foi realizado com o objetivo avaliar a capacidade de obtenção de boas soluções em tempo computacional reduzido fazendo uso da estrutura de redução de variáveis. Neste caso, partiu-se da solução inicial gulosa, utilizando a busca local e redução de variáveis com $tmax = (n - m / 2) * 0,2$ e um fator de redução $\Omega = 0,3$.

A utilização da estrutura de redução de variáveis oportunizou que o CPLEX explorasse a vizinhança com maior rapidez, permitindo a definição de um valor menor para o parâmetro *tmax*. Com estes parâmetros o problema da vizinhança não conseguir melhorar a solução inicial foi eliminado. Além disso, esta combinação proporcionou uma significativa redução no tempo computacional em comparação a resolução apenas com a busca local e aos tempos reportados em Vallada e Ruiz (2011), como pode ser observado na Tabela 15.

Tabela 15 – Resultados médios da busca local com redução

Tarefas	Máquinas	*DMR*	Melhor Sol (%)	Tempo-GA (s)	Tempo-Viz (s)
50	10	4,10	30,00	13	2,39
	15	-1,69	70,00	19	2,31
	20	-5,75	95,00	25	1,50
	25	-8,30	95,00	31	0,93
	30	-10,47	97,50	38	0,82
100	10	2,40	32,50	25	7,85
	15	-2,18	80,00	38	8,77
	20	-5,62	92,50	50	8,49
	25	-10,35	97,50	63	8,93
	30	-14,37	97,50	75	9,15
150	10	-1,38	72,50	38	16,68
	15	-3,28	85,00	56	16,46
	20	-7,15	95,00	75	18,43
	25	-12,47	100,00	94	19,12
	30	-14,48	100,00	113	18,85
200	10	-4,32	97,50	50	26,44
	15	-7,33	100,00	75	27,86
	20	-10,38	100,00	100	29,31
	25	-14,60	100,00	125	32,63
	30	-13,09	95,00	150	34,78
250	10	-5,39	100,00	63	41,01
	15	-9,65	100,00	94	41,58
	20	-12,23	100,00	125	47,50
	25	-16,71	100,00	156	47,55
	30	-18,96	100,00	188	53,75
Média		**-8,15**	**89,30**	**75**	**20,92**

Os resultados reportados na Tabela 15 mostram que o tempo computacional médio foi aproximadamente 3,5 vezes inferior ao reportado em Vallada e Ruiz (2011), o que é representativo mesmo considerando a diferença de *hardware*. Ainda, em 89,3% das instâncias foi possível igualar ou melhorar os resultados, mantendo uma média de 8,15% de melhora no valor de função objetivo encontradas pelo algoritmo genético.

A utilização da estratégia de redução de variáveis impede que parte da vizinhança seja explorada, evitando que determinadas soluções sejam obtidas, inclusive, em alguns casos, soluções ótimas ou que permitiriam melhorar na solução corrente. No entanto, ela permite uma significativa redução no tempo computacional.

Resultados da busca local + diversificação

A utilização da busca local + diversificação necessita da configuração de alguns parâmetros. O primeiro é o tempo máximo de execução *tmax*, definido como $tmax = (n - m / 2) * 0{,}2$. Já o parâmetro *tmin* é definido como $tmin = 0{,}01 * tmax$ no caso de uma iteração da busca local normal e $tmin = 0{,}2 * tmax$ no caso de uma iteração da busca + diversificação. Outro parâmetro a ser considerado é o IT_{max} que representa o número de vezes que a busca local com diversificação é utilizada. Neste caso, foi utilizado o valor fixo $IT_{max} = 10$. Como a utilização da redução de variáveis mostrou-se eficiente, a estrutura novamente foi utilizada nos testes, com um fator de redução $\Omega = 0{,}5$, e como solução inicial, fui utilizada a solução inicial gulosa aleatória. Os dados obtidos neste teste computacional são reportados na Tabela 16 mostrada a seguir.

Tabela 16 – Resultados médios da busca local com diversificação

Tarefas	Máquinas	*DMR*	Melhor Sol (%)	Tempo-GA (s)	Tempo-Viz (s)
50	10	0,85	55,00	13	42,84
	15	-4,61	87,50	19	26,23
	20	-7,34	95,00	25	11,62
	25	-9,95	97,50	31	7,12
	30	-11,64	97,50	38	3,34
100	10	-2,03	75,00	25	156,70
	15	-7,60	97,50	38	158,39
	20	-11,33	97,50	50	146,38
	25	-17,50	100,00	63	125,57
	30	-22,63	100,00	75	107,74
150	10	-5,71	97,50	38	264,42
	15	-9,57	100,00	56	246,37
	20	-13,54	100,00	75	250,12
	25	-20,11	100,00	94	229,31
	30	-23,40	100,00	113	241,61
200	10	-7,66	100,00	50	430,81
	15	-12,20	100,00	75	380,77
	20	-17,77	100,00	100	402,14
	25	-22,00	100,00	125	338,40
	30	-25,89	100,00	150	396,23
250	10	-9,19	100,00	63	535,61
	15	-15,53	100,00	94	579,04
	20	-18,98	100,00	125	542,19
	25	-24,52	100,00	156	541,76
	30	-27,77	100,00	188	503,61
Média		**-13,90**	**96,00**	**75**	**266,73**

Neste conjunto de testes, o foco principal foi a obtenção de soluções de boa qualidade, permitindo a utilização de uma amplitude maior no tempo computacional. Com isso, do conjunto de 1000 instâncias teste, foi possível igualar ou melhorar a solução em 960 casos, sendo estritamente menores em 923, com uma média do *DMR* de −13,90%.

Outros resultados a considerar

A utilização do CPLEX como um otimizador caixa preta permite que sejam ajustados diversos parâmetros que podem afetar a capacidade do otimizador em encontrar soluções ou o tempo de processamento. O CPLEX em sua versão 12.1 vem com o processamento em

paralelo em processadores com mais de um núcleo ativado por padrão. Como a máquina utilizada nos testes possui um processador com quatro núcleos, em determinados momentos da otimização, o trabalho é distribuído entre todos os núcleos. No entanto, ao contrário do esperado, um teste utilizando a busca local com a vizinhança de grande porte e o CPLEX utilizando o processamento sequencial, o tempo computacional foi aproximadamente 17,5% inferior ao tempo com processamento em paralelo. Além disso, a maior capacidade de exploração da vizinhança no mesmo tempo computacional permitiu que melhores soluções fossem encontradas, obtendo um *DMR* médio de −6,63 na execução sequencial, valor que é melhor que o da execução com quatro núcleos reportado na Tabela 14.

O desempenho superior da execução sequencial comparado com a execução paralela é atribuído possivelmente às estruturas de divisão de tarefas entre os núcleos de processamento aliado ao curto tempo da execução de uma resolução do modelo matemático da vizinhança de grande porte, embora os fatores reais não sejam conhecidos, já que o otimizador tem caráter comercial e funciona como caixa preta.

Mesmo com esta vantagem da execução sequencial, por padrão foi utilizado a execução em paralelo, já que o objetivo era manter a utilização do otimizador com a sua configuração padrão.

Outro teste realizado com a busca local + diversificação foi sua utilização partindo de um conjunto de soluções de melhor qualidade. Foram utilizadas como soluções iniciais as melhores soluções encontradas em alguns testes anteriores (não incluindo as soluções do teste de busca local + diversificação), tal que estas soluções iniciais possuíam um *DMR* médio de −12,93%. Com os mesmos parâmetros descritos nos resultados computacionais da busca local + diversificação, a busca foi capaz de melhorar 512 das 1000 instâncias, alterando o *DMR* médio para −15,19%.

Ao final de todos os testes computacionais realizados, do conjunto de 1000 instâncias de larga escala, foi possível obter soluções com valor de função objetivo inferior ou igual a obtida pelo algoritmo genético em 985 instâncias, sendo 962 estritamente melhores, proporcionando um *DMR* médio de −16,16%. Algumas destas soluções foram tão superiores aos encontrados pelo algoritmo genético que o *DMR* chegou a −44,55% para duas instâncias com 250 tarefas e 30 máquinas. Tanto as soluções quanto o conjunto de instâncias estão disponíveis para consulta no endereço http://www.inf.ufsm.br/~stefanello/research.

Apesar de na maioria dos casos o tempo computacional total médio ser superior ao tempo utilizado pelo algoritmo genético, vale lembrar que o objetivo principal deste trabalho é avaliar a capacidade de exploração da vizinhança. Além disso, originalmente, a vizinhança

foi proposta como um método de diversificação, ou pós-otimização, e aqui foi utilizada numa busca local e num algoritmo de múltiplos inícios, os quais requerem várias iterações.

Para uma avaliação mais justa e rigorosa da busca local + diversificação seria interessante dar mais tempo para o algoritmo genético para que este pudesse demonstrar sua capacidade de melhorar os resultados. No entanto, observa-se que utilizando a busca local com o método de redução, o tempo e a qualidade das soluções obtidas foram melhores que os reportados para o algoritmo genético.

5 CONCLUSÕES

5.1 Conclusões

O estudo apresentado neste trabalho refere-se a hibridização de métodos exatos e heurísticos para resolver dois problemas de otimização combinatória, sendo o primeiro o problema das p-medianas capacitado e o segundo o problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina com objetivo de minimizar o *makespan*. Foram propostas estratégias de hibridização distintas para resolver cada um dos problemas tratados. No PPMC foi utilizada a estratégia de eliminar heuristicamente variáveis do modelo matemático, além de algoritmo iterativo utilizado para exploração da vizinhança em determinadas regiões do conjunto de soluções através de um otimizador comercial. Já no problema de programação de tarefas em máquinas paralelas foi utilizado uma vizinhança de grande porte baseada em programação inteira mista, vizinhança esta utilizada internamente numa busca local e num algoritmo de múltiplos inícios.

No problema das p-medianas capacitado, após a apresentação formal do mesmo e uma breve revisão bibliográfica de trabalhos encontrados na literatura, foram apresentadas diferentes estratégias para resolução do problema. Foi proposta uma heurística de soluções inicias para obter e melhorar rapidamente uma solução inicial. Ainda, foram propostas quatro estruturas de redução de variáveis, cujo objetivo é eliminar heuristicamente variáveis do modelo matemático que dificilmente pertencerão a uma boa solução. Também foi proposto um algoritmo iterativo baseado nas reduções, que proporcionou a exploração da vizinhança a partir da análise de determinados subproblemas de forma iterativa na busca por alterações nos agrupamentos de modo a melhorar a solução corrente. Vários testes computacionais foram realizados a fim de comprovar a eficiência das estruturas de resolução proposta. Os testes computacionais foram realizados em cinco conjuntos de instâncias de modo a validar as estruturas de resolução proposta, as quais propiciaram a resolução das instâncias de pequeno, médio e, em especial, de grande porte. Os primeiros testes mostram a eficiência atual do otimizador com a utilização da restrição (4) do modelo matemático completo, de modo que, conforme o esperado, sua utilização proporcionou uma significativa redução no tempo

computacional relativo ao modelo completo. No que se refere às instâncias de pequeno e médio porte, para os quais a tecnologia atual resolve de forma exata a grande maioria das instâncias através do modelo completo, as estruturas de redução possibilitaram uma boa redução no tempo computacional com pouca perda na qualidade da solução em relação ao modelo completo. Para o conjunto 2 de instâncias as estratégias propostas neste trabalho se mostraram muito eficientes, pois foram capazes de encontrar as soluções ótimas em tempos computacionais notadamente inferiores, quando comparados aos melhores métodos da literatura da atualidade. No conjunto 4, além de encontrar soluções com valor de função objetivo melhor que as conhecidas, o tempo computacional também foi inferior. Já a utilização combinada de algumas reduções e o algoritmo iterativo proporcionou a resolução de instâncias de grande porte com uma quantidade de nós que ainda não haviam sido abordados na literatura.

Para o problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina com objetivo de minimizar o *makespan* foi proposta uma vizinhança de grande porte baseada em programação inteira mista, que é explorada com um otimizador comercial. Tal vizinhança foi utilizada em uma busca local e em um algoritmo de múltiplos inícios. Estruturas de redução de variáveis e um método de diversificação auxiliaram na exploração da vizinhança.

Os experimentos realizados demonstram que esta vizinhança é eficiente na obtenção de boas soluções para o problema. Em comparação com o algoritmo genético proposto em Vallada e Ruiz (2011), os resultados obtidos foram notadamente superiores para os conjuntos de instâncias considerados. Para as instâncias denominadas de pequeno porte, com a utilização do algoritmo de múltiplos inícios, foi possível encontrar soluções melhores ou iguais em todas as instâncias do conjunto. Para as instâncias de grande porte foi possível encontrar soluções melhores na grande maioria dos casos, de fato, considerando toda a pesquisa sobre esse assunto, os resultados obtidos são melhores, em média, 16,16% quando comparados aos de Vallada e Ruiz (2011).

5.2 Trabalhos futuros

Aplicar as estratégias de resolução utilizadas no problema das p-medianas capacitado em outros problemas correlatos, inclusive o não capacitado para o qual testes preliminares já indicaram ser viável e competitivo.

No referido problema de programação de tarefas em máquinas paralelas, pretende-se observar o comportamento da vizinhança como um método de intensificação e diversificação internos a alguma metaheurística como, por exemplo, busca tabu. Além disso, pretende-se aplicar esta vizinhança a problemas similares encontrados na literatura, e ainda inclusão de planos de cortes no modelo matemático. Também pretende-se avaliar novas estratégias de redução de variáveis bem como utilizar estratégias no método de diversificação que evitem o retorno a ótimos locais já conhecidos permitindo uma exploração ainda mais eficiente e rápida da vizinhança. Ainda, é possível projetar um algoritmo exato que aproveite características inerentes ao modelo matemático utilizado para explorar a vizinhança de grande porte, ao invés de lançar mão de otimizadores do tipo caixa preta.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMADI, S.; OSMAN, I. H. Greedy random adaptive memory programming search for the capacitated clustering problem. **European Journal of Operational Research**, v. 162, n. 1, p. 30-44, 2005.

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. **Omega-International Journal of Management Science**, v. 27, n. 2, p. 219-239, 1999.

ANBIL, R.; GELMAN, E.; PATTY, B.; TANGA, R. Recent Advances in Crew-Pairing Optimization at American Airlines. **Interfaces**, v. 21, n.1, p. 62-74, 1991.

ARMENTANO, V. A.; DE FRANÇA, M. F. Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based GRASP approach. **European Journal of Operational Research**, v. 183, n.1, p. 100-114, 2007.

ARORA, S.; BARAK, B. **Computational complexity: a modern approach**. Cambridge, New York, Cambridge University Press, 2009.

BALDACCI, R.; HADJICONSTANTINOU, E.; MANIEZZO, V.; MINGOZZI, A. A new method for solving capacitated location problems based on a set partitioning approach. **Computer & Operations Research**, v. 29, n. 4, p. 365-386, 2002.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Computing Surveys**, v. 35, n. 1, p. 268-308, 2003.

BOCCIA, M.; SFORZA, A.; STERLE, C.; VASILYEV, I. A Cut and Branch Approach for the Capacitated p-Median Problem Based on Fenchel Cutting Planes. **Journal of Mathematical Modelling and Algorithms**, v.7 n. 1, p. 43-58, 2008.

CHAVES, A. A.; CORREA, F.A.; LORENA, L. A. N. Clustering Search Heuristic for the Capacitated p-Median Problem. **Advances in Software Computing Series,** v. 44, p. 136-143, 2007.

CHEN, J. F. Minimization of maximum tardiness on unrelated parallel machines with process restrictions and setups. **International Journal of Advanced Manufacturing Technology**, v. 29, n. 5, p. 557-563, 2006.

CHEN, J. F.; WU, T. H. Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. **Omega-International Journal of Management Science**, v. 34, n. 1, p. 81-89, 2006.

DE PAULA, M. R.; RAVETTI, M. G.; MATEUS, G. R.; PARDALOS, P. M. Solving parallel machines scheduling problems with sequence-dependent setup times using Variable Neighbourhood Search. **IMA Journal of Management Mathematics**, v.18, n. 2, p. 101-115, 2007.

DIAZ, J.A., FERNANDEZ, E. Hybrid scatter search and path relinking for the capacitated p-median problem. **European Journal of Operational Research**, v. 169, n. 2, p. 570-585, 2006.

DUMITRESCU, I.; STÜTZLE, T. Combinations of local search and exact algorithms. **Applications of Evolutionary Computing**, LNCS, v. 2611, p. 211-223, 2003.

FANJUL-PEYRO, L.; RUIZ, R. Size-reduction heuristics for the unrelated parallel machines scheduling problem. **Computers & Operations Research**, v. 38, n. 1 , p. 301-309, 2011.

FERNANDES S.; LOURENÇO, H. R. Optimised Search Heuristics: A Mapping of procedures and Combinatorial Optimisation Problems, 2008. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.2032>. Acesso em: 9 jan. 2011.

FLESZAR, K.; HINDI, D. S. An effective VNS for the capacitated p-median problem. **European Journal of Operational Research**, v. 191, n. 3, p. 612-622, 2008.

FRANÇA, P.; GENDREAU, M.; LAPORTE, G.; MÜLLER, F. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. **International Journal of Production Economics**, v. 43, n. 2-3, p79-89, 1996.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability: a GUIDE to the theory of NP-completeness**. W. H. Freeman, 1979.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G. Optimization and approximation in deterministic sequencing and scheduling: A survey. **Annals of Discrete Mathematics**, v. 5, p. 287-326, 1979.

HAKIMI, S.L. Optimal locations of switching centers and the absolute centers and the medians of a graph. **Operations Research**, v. 12, n. 3, p. 450-459, 1964.

JOURDAN, L.; BASSEUR, M.; TALBI, E.G. Hybridizing exact methods and metaheuristics: A taxonomy. **European Journal of Operational Research,** v. 199, n. 3, p. 620-629, 2009.

KIM, D.W.; KIM, K.H.; JANG, W.; CHEN, F.F. Unrelated parallel machine scheduling with setup times using simulated annealing. **Robotics and Computer-Integrated Manufacturing**, v. 18, n. 3-4, p. 223-231, 2002.

LOGENDRAN, R.; MCDONELL, B.; SMUCKER, B. Scheduling unrelated parallel machines with sequence-dependent setups. **Computers & Operations Research**, v. 34, n. 11, p. 3420-3438, 2007.

LORENA, L. A. N.; PEREIRA, M. A.; SALOMÃO, S. N. A. A relaxação lagrangeana/surrogate e o método de geração de colunas: novos limitantes e novas colunas. **Revista Pesquisa Operacional**, v.23, n. 1, p. 29-47, 2003.

LORENA, L. A. N.; SENNE, E. L. F. Abordagens de Geração de Colunas para um Problema de p-medianas Capacitado. **XXXIV SBPO – Simpósio Brasileiro de Pesquisa Operacional,** Rio de Janeiro, 2002.

LORENA, L. A. N.; SENNE, E. L. F. A column generation approach to capacitated p-median problems. **Computers & Operational Research**, v. 31, n. 6, p. 863-876, 2004.

LOW, C. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. **Computers & Operations Research**, v. 32, n. 8, p. 2013-2025, 2005.

MANIEZZO, V.; MINGOZZI, A.; BALDACCI, R. A bionomic approach to the capacitated p-median problem. **Journal of Heuristics**, v. 4, n. 3, p. 263-280, 1998.

MAUTOR, T.; MICHELON, P. "MIMAUSA: A new hybrid method combining exact solution and local search". **Proceedings of the 2nd International Conference on Metaheuristics**, p. 15. Sophia-Antipolis, France, 1997.

MAUTOR, T.; MICHELON, P. "MIMAUSA: an application of referent domain optimization". **Technical Report**, 260, Laboratoire d'Informatique, Universit´e d'Avignon et des Pays de Vaucluse, 2001.

MLADENOVIC, N.; e HANSEN, P. Variable Neighborhood Search. **Computers & Operations Research**, v. 24, n. 11, p. 1097-1100, 1997.

MULVEY, J.M.; BECK, M.P. Solving capacitated clustering problems. **European Journal of Operational Research**, v. 18, n. 3, p 339-348, 1984.

NEMHAUSER, G. L.; WOLSEY, L. **Integer and Combinatorial Optimization**. New York, John Wiley & Sons, 1988.

NIEVERGELT, J. Exhaustive Search, Combinatorial Optimization and Enumeration: Exploring the Potential of Raw Computing Power. **SOFSEM 2000: Theory and Practice of Informatics**, LNCS, v. 1963, p. 18-35, 2000.

OSMAN, I. H.; CHRISTOFIDES, N. Capacitated clustering problems by hybrid simulated annealing and tabu search. **International Transactions in Operational Research**, v. 1, n. 3, p. 317-336, 1994.

PINEDO, M. **Scheduling: Theory, Algorithms, and Systems**. 3th ed, New Jersey: Prentice Hall, 678 p, 2008.

PUCHINGER, J.; RAIDL, G.R. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. **International Work-Conference on the Interplay Between Natural and Artificial Computation - IWINAC, Part II**. LNCS 3562, p. 41–53, 2005.

RABADI, G.; MORAGA, R.; SALEM, A. Heuristics for the unrelated parallel machine scheduling problem with setup times. **Journal of Intelligent Manufacturing**, v. 17, n. 1, p. 85-97, 2006

SCHEUERER, S.; WENDOLSKY, R. A scatter search heuristic for the capacitated clustering problem. **European Journal of Operational Research**, v. 169, n. 2, p. 533-547, 2006.

STEFANELLO, F.; MÜLLER, F. M. Um Estudo Sobre Problemas de Agrupamento Capacitado. **Anais XLI SBPO – Simpósio Brasileiro de Pesquisa Operacional**, Porto Seguro, 2009.

STEFANELLO, F.; MÜLLER, F..M.; ARAÚJO, O.C.B. Problema das p-Medianas Capacitado: Estudo Sobre a Resolução de Instâncias por Métodos Exatos. **Anais IX SEPROSUL: Semana de Engenharia de Produção Sul-Americana**, Piriápolis – Uruguai, 2009a.

STEFANELLO, F.; MÜLLER, F. M.; ARAÚJO, O.C.B. Estudo sobre a Resolução do Problema das P-Medianas Capacitado. **Anais III ERPOSul: Encontro Regional de Pesquisa Operacional da Região Sul,** Porto Alegre, v. 1, p. 31-35, 2009b.

TALBI, E. G. A taxonomy of hybrid metaheuristics, **Journal of Heuristics**, v. 8, n. 5, p. 541-564, 2002.

VALLADA, E.; RUIZ, R. Genetic algorithms for the unrelated parallel machine scheduling problem with sequence dependent setup times. Aceito para publicação em: **European Journal of Operational Research**, 2011.

YU, L. et al. Scheduling of unrelated parallel machines: an application to PWB manufacturing, **IIE Transactions**, v. 34, n. 11, p. 921-931, 2002.

WENG, M. X.; LU, J.; REN, H. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. **International Journal of Production Economics**, v. 70, n. 3, p. 215-226, 2001.