PARALELIZAÇÃO DA METAHEURÍSTICA *ITERATED GREEDY SEARCH* APLICADA AO PROBLEMA DE INSTALAÇÃO DE FIBRAS EM REDES ÓTICAS

Daniel Morais dos Reis, **Álvaro Martins Espíndola**, **Guilherme de Morais Bessa**

danielmorais@div.cefetmg.br, alvaromaresp@gmail.com, guilherme.morais.bessa@gmail.com

Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

Divinópolis, MG, 35503-822, Brazil

Sérgio Ricardo de Souza, **Anolan Yamilé Milanés Barrientos**

sergio@dppg.cefetmg.br, anolan@decom.cefetmg.br

Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

Belo Horizonte, MG, 30510-000, Brazil

Resumo. *Técnicas de paralelismo, além da levarem à execução acelerada de programas, permitem também que estes sejam implementados de diferentes formas, as quais obtém desempenhos diferentes entre si não somente sob a ótica da redução do tempo final de execução necessário, mas também também acerca da qualidade de solução encontrada. O presente artigo avalia qual, dentre diferentes formas de implementações paralelas da metaheurística* Iterated Greedy Search *(IGS), otimiza o seu desempenho ao ser aplicada à solução de instâncias do Problema de Instalação de Fibras em Redes Óticas - PIFRO, quando realizada com técnicas de paralelismo em uma mesma CPU dotada com mais de um núcleo de processamento em um ambiente de memória compartilhada. O desempenho é calculado através da verificação do* speedup *encontrado à medida em que são disponibilizadas mais* threads *executadas simultaneamente e um valor alvo a ser atingido. A paralelização da metaheurística IGS é realizada nas modalidades de Paralelização Mono-perturbação e Paralelização Multiperturbação, sendo ambas testadas com e sem a sincronização da melhor solução corrente. O* speedup *encontrado entre as diferentes formas de paralelização mostra-se superlinear em alguns casos.*

Palavras chave: *Problema de Instalação de Fibras Óticas, Paralelismo, Metaheurísticas*

1 INTRODUÇÃO

A execução simultânea de tarefas através do particionamento do trabalho entre diversos núcleos de processamento permite que sejam encontradas soluções em um menor tempo do que na execução sequencial e que instâncias de maior dimensão dos problemas possam ser exploradas. O emprego de técnicas de paralelismo para solucionar problemas de otimização permite combinar a execução simultânea de tarefas de diferentes formas em diferentes espaços ou subespaços de busca, obtendo, assim, soluções com melhor qualidade.

Estratégias de paralelismo podem ser aplicadas tanto a métodos exatos quanto a métodos heurísticos/metaheurísticos. Diferentemente dos métodos exatos, que possuem como principal objetivo alcançar a solução ótima, as heurísticas/metaheurísticas ao serem paralelizadas precisam avaliar o quão distante está a solução retornada da solução ótima e o quão rápido essas soluções são retornadas, afim de se obter assim uma análise de seu custo/benefício de utilização. As estratégias de paralelismo são um recurso muito bem aproveitado pelas heurísticas/metaheurísticas e novas formas de paralelismo são necessárias para o avanço da literatura [Crainic et al., 2014].

Apresentada por [Lourenço et al., 2010], a metaheurística Busca Gulosa Iterativa (*Iterated Greedy Search* – IGS) parte de uma solução inicial para um determinado problema de otimização combinatória para, em seguida, fazer buscas locais e melhorar por diversas vezes a solução inicial encontrada através de um determinado número de iterações. A boa qualidade das soluções encontradas pela metaheurística IGS em [Garcia et al., 1998], [Fanjul-Peyroa and Ruiz, 2010], [Gagnaire and Doumith, 2007], [Ruiz and Stützle, 2007], [Ruiz and Stützle, 2008], [Framinan and Leisten, 2008] e [Goulart et al., 2011], dentre outras, em que foram apresentadas implementações de forma sequencial, motiva neste trabalho sua implementação de diferentes formas paralelizadas, com diferentes estratégias que modificam seu funcionamento e, consequentemente, a qualidade final de suas soluções e o tempo para que estas sejam atingidas.

O Problema de Instalação de Fibras em Redes Óticas - PIFRO (*Fiber Installation Problem* - FIP), apresentado inicialmente em [Antonakopoulos and Zhang, 2007], consiste em alocar todos os comprimentos de onda requisitados em uma rede de fibras óticas de forma a reduzir a quantidade de equipamentos necessários, provendo, assim, a redução do seu custo final de instalação. Em [dos Reis et al., 2013], a metaheurística IGS, em uma implementação sequencial, é utilizada com sucesso como estratégia de solução para o problema, encontrando as melhores soluções conhecidas na literatura. Estes fatores fazem com que PIFRO e algumas de suas instâncias mais complexas sejam bons candidatos para averiguação da eficiência da aplicação das estratégias de paralelização propostas para IGS.

O artigo está organizado como segue: a Seção 2 descreve o funcionamento e estrutura de algoritmos paralelos; a Seção 3 descreve a metaheurística IGS e as estratégias de paralelização aqui abordadas; a Seção 4 descreve PIFRO, sua caracterização e equipamentos utilizados; a Seção 5 detalha as estratégias de solução já abordadas na literatura para PIFRO; as Seções 6, 7 apresentam e detalham os resultados obtidos respectivamente; a Seção 7 exibe as referências utilizadas por este trabalho.

2 ESTRUTURA E ANÁLISE DO DESEMPENHO DE ALGORITMOS PARALELOS

A divisão do trabalho entre algoritmos paralelos é realizada pela atribuição de tarefas que podem ser executadas simultaneamente em diferentes núcleos de processamento. Uma vez concluída a execução dessas tarefas, seus dados são unidos de forma a compor soluções, as quais devem ou não serem aceitas após filtradas por critérios previamente determinados. Segundo [Hill and Marty, 2008a], a partir do tempo de execução de um algoritmo sequencial pode-se estimar o tempo de execução de um algoritmo paralelo através de:

$$Tempo_{paralelo} = \frac{Tempo_{sequencial}}{p} \tag{1}$$

em que $Tempo_{paralelo}$ indica o tempo esperado para a execução de um algoritmo em p núcleos de processamento e $Tempo_{sequencial}$ é o tempo de execução do mesmo algoritmo em apenas um núcleo de processamento. O tempo paralelo neste caso é válido somente para algoritmos que possuem todo o seu código completamente paralelizado, o que, via de regra, é difícil, pois existem tarefas que não podem ser realizadas em paralelo, como, por exemplo, operações sobre uma variável global por vários núcleos de processamento simultaneamente, o carregamento de dados iniciais para o algoritmo, dentre outras.

A lei de Amdahl [Hill and Marty, 2008b, Sun and Chen, 2010, Amdahl, 1967] mostra que, a partir do percentual de código paralelizado, pode-se saber o tempo de execução que o algoritmo paralelo pode precisar. Para este cálculo, a variável $perc_p$ representa o percentual de código paralelizado de um algoritmo e a variável $perc_{np}$ o percentual de código sequencial, ambos com um valor entre 0 e 1. O tempo paralelo mínimo para execução de um algoritmo paralelo é dado por:

$$Tempo_{paralelo} = perc_p.\frac{Tempo_{sequencial}}{p} + Tempo_{sequencial}.perc_{np} \tag{2}$$

Na literatura, a forma de mensuração mais importante do desempenho de algoritmos paralelos é dado pelo *Speedup*. O resultado desta métrica é obtido relacionando-se os tempos de execução sequenciais e paralelos [Alba, 2005]. O valor do *Speedup* de um algoritmo paralelo é obtido por:

$$Tempo_{paralelo} = \frac{Tempo_{sequencial}}{Tempo_{paralelo}} \tag{3}$$

Segundo [Bader et al., 2002, Pacheco, 2011], algoritmos que pesquisam soluções aleatoriamente (como, por exemplo, heurísticas/metaheurísticas) em determinados espaços/subespaços de busca de tamanhos variáveis ou não, podem apresentar anomalias em seu *Speedup*, dado que, dependendo da disposição das soluções e da pesquisa feita pelo algoritmo, a adição ou exclusão de núcleos de processamento gera um aumento ou redução de desempenho também aleatórios advindos da diferente quantidade de instruções executadas. Sendo assim, é normal que estes algoritmos possam apresentar *Speedup* maior ou menor que o estimado pela Equação 3. No entanto, por este motivo, não violam a lei de Amdahl, mesmo quando a estes é dado o melhor caso possível para o algoritmo sequencial e um caso que não o melhor para o algoritmo paralelo [Bader et al., 2002]. Caso o *Speedup* de um algoritmo seja superior ao tempo estimado pela Equação 3, este *Speedup* será

início IGS()
1. $s \leftarrow$ Construção();
2. $s, s^* \leftarrow$ BuscaLocal(s);
3. **enquanto** critério de parada não for satisfeito **faça**
4. $\quad s^{'} \leftarrow$ Perturbação(s);
5. $\quad s^{'} \leftarrow$ BuscaLocal($s^{'}$);
6. $\quad$ **se** $s^{'}$ é melhor que s^* **então** $s^* \leftarrow s^{'}$;
7. $\quad$ **se** CritérioDeAceitação $(s, s^{'}, s^*)$ **então** $s \leftarrow s^{'}$;
8. **fim-enquanto**;
9. **retorne** s^*;

fim

Figura 1: Pseudocódigo da metaheurística IGS.

chamado de superlinear e, caso contrário, será chamado de sublinear [Pacheco, 2011]. A análise desse fator é a principal motivação deste trabalho.

A divisão lógica do trabalho em diferentes núcleos de processamento pode ser feita através de estruturas chamadas *threads*. Assim sendo, teoricamente, à medida em que se possui mais *threads* dividindo o trabalho total do algoritmo, menor será seu tempo final de execução. As arquiteturas atuais dos processadores permitem que cada um de seus núcleos execute mais de uma *thread* simultaneamente através do aproveitamento de estruturas ociosas do núcleo quando estão disponíveis (*Simultaneous multithreading* - SMT)[Tullsen et al., 1995].

3 ESTRATÉGIAS DE SOLUÇÃO PARALELAS

O pseudocódigo do metaheurística IGS é exibido no Algoritmo 1. Na linha 1, heurísticas construtivas para o problema são utilizadas para gerar uma solução inicial S na qual, posteriormente na linha 2, é feita uma busca local, resultando na solução intermediária $S^{'}$ e, respectivamente, na melhor solução encontrada até o momento. Em seguida, enquanto o critério de parada da linha 3 não for satisfeito, é aplicada uma perturbação na linha 4, descontruindo parcialmente a solução, removendo-se partes dela aleatoriamente, para, em seguida, adicioná-las novamente em uma nova ordem aleatoriamente definida, produzindo, portanto, uma solução diferente. Na linha 5, é realizada novamente uma busca local na solução $S^{'}$ tentando melhorá-la. A linha 6 garante que a melhor solução encontrada até o momento pelo algoritmo seja armazenada em S^*. Para escapar de ótimos locais, na linha 7 do algoritmo, a solução S pode ser modificada para aceitar uma solução diferente da melhor, através de um diferente critério de aceitação [Lourenço et al., 2010], desde que ela esteja dentro os limites aceitáveis quando comparada a s^*. Por este motivo e para que a melhor solução encontrada até o momento (S^*) não seja perdida, na perturbação e na busca local da metaheurítisca (linhas 4 a 5) a solução s é utilizada.

3.1 PARALELIZAÇÃO DA METAHEURÍSTICA IGS

Neste trabalho, a metaheurística IGS-Paralelo recebe os seguintes diferentes modos de execução:

(i) Paralelização Monoperturbação/Multiperturbação: inicialmente, a metaheurística gera uma solução para cada *thread* solicitada, as quais deverão sofrer construções, perturbações e buscas locais no decorrer de sua execução. O valor da perturbação que cada *thread* faz em suas soluções pode ser o mesmo para todas as *threads* disponíveis ou um valor específico de perturbação para cada uma;

(ii) Sincronização da melhor solução corrente: este parâmetro faz com que quando alguma *thread* consiga uma melhor solução, esta seja copiada para as demais *threads*, substituindo suas soluções correntes.

O pseudocódigo de IGS-Paralelo é exibido no Algoritmo 2.

Inicialmente, a metaheurística IGS-Paralelo recebe, como parâmetros, os dados relevantes ao seu funcionamento, como a quantidade de *threads* que serão disponibilizadas para a metaheurística, dois valores booleanos, que indicam, respectivamente, se deve haver sincronização de soluções e diferentes perturbações para cada solução utilizada em cada *thread*.

Entre as linhas 1 a 4, em paralelo, cada *thread* tem sua respectiva solução inicializada e realizada uma busca local. Na linha 5, a melhor solução encontrada até o momento é armazenada nas variáveis, que representará posteriormente a melhor solução encontrada até o momento por cada respectiva *thread*. Entre as linhas 6 a 10 é definida uma sessão crítica onde sequencialmente cada *thread* respectivamente verifica se a sua melhor solução é a melhor solução encontrada dentre todas as *threads* e, em caso afirmativo, a armazena na solução S^*_{global}. Enquanto um critério de parada pré-definido não for alcançado, o laço entre as linhas 11 e 33 será executado. Entre as linhas 12 a 17, de acordo com a quantidade de *threads* especificada, uma perturbação será aplicada na solução temporária de cada *thread*. Caso esteja habilitada a opção de multiperturbação, na linha 14, cada *thread* irá aplicar a perturbação com um valor diferente; caso contrário, na linha 16, cada *thread* aplicará a perturbação com o mesmo valor na respectiva solução temporária utilizada. Na linha 18, para cada solução temporária de cada *thread*, é executada uma busca local, na tentativa de melhorar a solução que veio da perturbação anteriormente executada. Nas linhas 19 a 20, caso alguma das respectivas soluções encontradas seja, respectivamente melhor, que a melhor solução encontrada por sua respectiva thread, esta será atualizada. Nas linhas 21 a 29, caso esteja habilitada a opção de sincronização, todas as *threads* comparam suas respectivas melhores soluções encontradas com a melhor dentre todas as soluções (S^*_{global}), fazendo com que assim, então, ela passe a ser utilizada como nova solução temporária para cada *thread* na próxima iteração do laço. Isso deve ser tratado como condição de corrida uma vez que várias *threads* podem tentar simultaneamente alterar tal valor e é por isso que entre as linhas 22 a 26 uma sessão crítica é definida. Após a sincronização entre as *threads* executada na linha 27 a melhor solução dentre todas é compartilhada entre todas as *threads* na linha 27. Entre as linhas 31 e 33, para escapar de ótimos locais, a solução temporária de cada *thread* pode ser atualizada e utilizada nas próximas iterações do laço, de acordo com um critério de aceitação previamente configurado, desde que ela esteja dentro os limites aceitáveis quando comparada a s^*_{global}. Na linha 36, a metaheurística IGS-Paralelo retorna a melhor solução global encontrada.

4 PIFRO COMO INSTÂNCIA DE TESTE DA METAHEURÍSTICA IGS-PARALELO

Em [dos Reis et al., 2013] e [Reis et al., 2012], a metaheurística IGS foi utilizada como estratégia de solução para o Problema de Instalação de Fibras em Redes Óticas (PIFRO), obtendo os melhores resultados conhecidos, no entanto, sem o emprego de técnicas de paralelismo. Isso faz com que PIFRO seja uma boa instância de testes para a metaheurística IGS-Paralelo aqui apresentada. Baseando-se nos resultados obtidos em [dos Reis et al., 2013] e [Reis et al., 2012], são aplicadas estratégias de paralelismo a IGS-PIFRO em arquiteturas de processadores *multicore* e mostra-se que ela pode atingir valores alvos de custos para instâncias de redes óticas do PIFRO cada vez mais rapidamente, à medida em que são disponibilizados mais *threads* para a execução de tarefas com uma taxa acima da esperada inicialmente (*Speedup* superlinear).

As redes óticas de PIFRO, em cada um dos seus enlaces, através de fibras óticas, podem transmitir dados em uma ordem de terabits por segundo. Esta velocidade é muito maior que a dos dispositivos utilizados para recepção e transmissão de dados. O custo de um *backbone* deste tipo é de centenas de milhões de dólares e, por isso, qualquer percentual de redução neste valor torna-se significativo. A Tecnologia de Divisão e Multiplexação de Comprimentos de Onda (WDM - *Wavelength Division Multiplexing*) permite que seja utilizada toda a capacidade de transmissão de dados de uma fibra ótica, dividindo-a logicamente em vários canais que podem ser utilizados simultaneamente. Caminhos Óticos são conexões entre uma origem e um destino sem a conversão do sinal do domínio ótico para o domínio elétrico. O custo diferente para cada comprimento de onda adicionado em um enlace de uma rede ótica proporciona uma alta economia no custo final de instalação da rede.

4.1 CARACTERIZAÇÃO DO PROBLEMA

Em [dos Reis et al., 2013] e [Reis et al., 2012], PIFRO é definido por um grafo não direcionado $G = (V, E)$, o qual representa a topologia física da rede. O conjunto V representa o conjunto de nós da rede, ou seja, os chaveadores óticos, e E representa o conjunto de fibras óticas que conectam estes nós. Cada aresta $[i, j] \in E$ está associada a uma distância em quilômetros entre os nós conectados a suas extremidades. Seja $R = r_1, r_2, ..., r_n$ o conjunto de requisições por caminhos óticos, de modo que cada requisição é definida por: (i) uma origem; (ii) um destino; e (iii) uma quantidade de comprimentos de onda a serem alocados em um caminho ótico; PIFRO consiste então, em encontrar caminhos óticos para as requisições R em uma rede ótica de modo a minimizar o custo de instalação de equipamentos da rede. Este problema foi proposto em [Antonakopoulos and Zhang, 2007] como um esforço da Bell Labs em desenvolver métodos de otimização para redes óticas. No presente trabalho, é considerado o custo da rede correspondente aos três equipamentos óticos mais caros utilizados em redes óticas. O objetivo de PIFRO é minimizar o custo de instalação de equipamentos da rede os quais são descritos a seguir.

ROADM

O ROADM (*Reconfigurable Optical Add/Drop Multiplexer*) faz o chaveamento ótico das requisições R sem a necessidade de conversão do sinal do domínio ótico para o domínio

elétrico e vice versa. Podem ser adicionadas k portas em um ROADM e, em cada porta, conectada uma fibra ótica WDM com capacidade para transmitir $\mu = 100$ comprimentos de onda. Na medida em que são necessárias, novas portas podem ser instaladas a um custo de uma fração de milhão de dólares, o que tornará seu custo linearmente proporcional. A modelagem do custo total de utilização de equipamentos ROADM na rede ótica é calculada somando-se o custo de duas portas ROADM por cada fibra ótica instalada na rede.

Seja então $ROADM_{COST}$ o custo de uma porta ROADM. O custo total de ROADM por fibra ótica em um enlace da rede é definido por:

$$c_1 = 2.ROADM \tag{4}$$

OT

O segundo equipamento em termos de custo a ser descrito é o OT (*Optical Transponder*). Ele faz a conversão de sinais do domínio ótico para o domínio elétrico e do domínio elétrico para o domínio ótico. Um OT é instalado apenas em locais da rede onde há a necessidade de conversão de sinais. Seja, então, L_{OT} a distância em cada fibra ótica em que deverá ser instalado um OT e seja OT_{COST} o custo de instalação na rede ótica de cada OT da ordem de 5 a 10% do valor de uma porta ROADM. Seja também l o comprimento de uma fibra ótica. Assim, o custo total de equipamentos OT por fibra ótica em cada enlace da rede é dado por:

$$c_3 = \frac{1}{L_{OT}}.2.OT_{COST} \tag{5}$$

OPTICAL AMPLIFIER (OA)

A dispersão cromática é um problema que causa a diminuição de potência do sinal ótico, impedindo-o, assim, de chegar ao seu destino algumas vezes. Assim, quando for necessária amplificação do sinal ótico para compensar sua dispersão, deverá ser instalado um equipamento OA (*Optical Amplifier with Dispersion Compensation*).

Seja então L_{OA} a distância em cada fibra ótica em que deve ser instalado um equipamento OA, e seja OA_{COST} o custo de instalação de cada equipamento OA. Desse modo, o custo total de instalação de equipamentos OA em cada fibra ótica por enlace da rede é definido por:

$$c_2 = \frac{1}{L_{OA}}.OA_{COST} \tag{6}$$

CUSTO TOTAL DA REDE

O custo total da rede é calculado pelo valor da função F_{WDM} para cada enlace, dado por:

$$F^{ij}_{WDM}(\omega_{ij}, l_{ij}) = c_1 \left\lceil \frac{\omega_{ij}}{\mu} \right\rceil + c_2 l_{ij} \left\lceil \frac{\omega_{ij}}{\mu} \right\rceil + c_3 \omega_{ij} l_{ij} \tag{7}$$

Nesta função, w é a quantidade de comprimentos de onda a serem alocados em um determinado enlace da rede; μ é o número de comprimento de onda a serem transmitidos; é

o comprimento de uma fibra ótica; e c_1, c_2 e c_3 são as constantes definidas pelas expressões 4, 5, 6, representando os custos de instalação dos equipamentos ROADM, OT e OA, respectivamente.

A representação gráfica do comportamento desta função em termos do número de comprimentos de onda a serem transmitidos é mostrada na Figura 3. Note o caráter linear por partes da função $F_{WDM}(w, l)$.

5 REVISÃO BIBLIOGRÁFICA DO PIFRO

Segundo [Antonakopoulos and Zhang, 2007], os subproblemas de roteamento e atribuição de comprimentos de onda devem ser tratados separadamente. Eles sugerem que as heurísticas para PIFRO devem focar no problema de roteamento, pois o subproblema de atribuição de comprimentos de onda pode ser resolvido como um problema de coloração de grafos em um grafo de conflitos [Hertz and de Werra, 1987, Hyytiä and Virtamo, 1998, Noronha and Ribeiro, 2006]. Em [Antonakopoulos and Zhang, 2007], foram propostas duas heurísticas construtivas para PIFRO, nas quais foi verificado que a ordem na qual as requisições R são alocadas na rede ótica influencia no custo final da rede. Para a criação de um caminho ótico, vários ROADM devem ser utilizados, conectados entre si para formar o caminho ótico desejado. Um salto é cada par de portas ROADM utilizadas para compor um caminho ótico.

Em [Antonakopoulos and Zhang, 2007] foram avaliadas heurísticas recebendo, como parâmetro, requisições ordenadas por: (a) ordem na qual aparecem na entrada; (b) ordenadas de acordo com a quantidade de saltos entre os ROADMs em ordem crescente e decrescente; (c) ordenadas por menor quantidade de comprimentos de onda a serem alocados na rede; e (d) ordenadas de forma aleatória, atribuindo-se um valor aleatório a cada requisição $r \in R$ e colocando-as em ordem ascendente, de acordo com este valor. A ordenação (d) foi a que produziu melhores resultados.

A heurística *Greedy*, proposta em [Antonakopoulos and Zhang, 2007], aborda um procedimento guloso, no qual, para cada requisição de $r \in R$ a ser alocada na rede ótica, é calculado o seu custo incremental, dado por:

$$C(i) = F_{WDM}(\omega_i + \alpha, l) - F_{WDM}(\omega_i, l) \tag{8}$$

sendo α o número de comprimentos de onda a serem adicionados para todos os enlaces da rede. Em seguida, é executado o algoritmo Dijkstra para encontrar o caminho ótico com menor custo incremental na rede para a requisição. Os enlaces selecionados pelo algoritmo Dijkstra têm, adicionado a seu custo, o custo incremental calculado.

Após encontrado o caminho com menor custo incremental para todas as requisições $r \in R$, é feito o refinamento iterativo, que é uma busca local do tipo 1-reroteamento, em que o custo incremental de uma requisição é removido dos enlaces definidos para formarem o seu caminho ótico na rede ótica e cada uma das demais requisições $r \in R$ fazem o mesmo, porém, novamente adicionando o seu custo incremental a um novo caminho ótico com o menor custo incremental, se possível, ou utilizando o caminho ótico já existente, caso contrário.

Este processo é realizado para cada requisição $r \in R$ alocada na rede ótica, até que não seja mais possível encontrar nenhum caminho ótico com custo incremental menor

para nenhuma requisição na rede ótica. Ao final, a primeira requisição que teve seu custo incremental removido do caminho ótico associado a ela da rede ótica será novamente alocada em um novo caminho ótico com o menor custo incremental possível, ou em seu caminho ótico já definido anteriormente, caso contrário.

A heurística PSC, também proposta em [Antonakopoulos and Zhang, 2007], é semelhante à heurística *Greedy*. No entanto, utiliza uma função de custo gulosa diferente. Enquanto a heurística Greedy utiliza a função F_{WDM}, a heurística PSC utiliza uma função de custo gulosa mais sofisticada, dada por:

$$F'_{WDM}(\omega, l, p, P) = (c_1 + c_2 l)\left[\frac{p}{P}\left\lceil\frac{\omega}{\mu}\right\rceil + \frac{P-p}{P} f\left(\frac{\omega}{\mu}\right)\right] + c_3\omega l \tag{9}$$

Na expressão 9, p é o passo corrente do algoritmo; P é um valor inteiro; e:

$$f(x) = \lfloor x \rfloor + \sqrt{x - \lfloor x \rfloor} \tag{10}$$

No primeiro passo do algoritmo de busca local (refinamento iterativo) da heurística PSC, p é igual a zero. Neste momento, a função de custo é equivalente a $f(\frac{w}{\mu})$. A cada iteração da heurística sobre o conjunto de requisições $r \in R$, o valor de p é incrementado. Desta forma, F_{WDM} converge gradualmente para F'_{WDM}. A velocidade da convergência é definida pelo parâmetro P, que foi definido como $P = 5$, através de experimentos apresentados em [Antonakopoulos and Zhang, 2007]. A função $F'_{WDM}(w, l, p, P)$ está apresentada na Figura 4.

Em [dos Reis et al., 2013], as heurísticas *Greedy* e PSC, apresentadas inicialmente em [Antonakopoulos and Zhang, 2007], são utilizadas na construção de soluções para IGS e o refinamento iterativo proposto também em [Antonakopoulos and Zhang, 2007] utilizado como busca local.

6 EXPERIMENTOS

Neste trabalho foi avaliado o desempenho da heurística IGS-PIFRO-Paralelo utilizando em algumas instâncias da biblioteca Survivable Network Design Library - SNDlib, disponível em [Zuse-Institute,]. SNDlib é uma biblioteca de instâncias de testes para problemas de otimização em redes [Gonçalves and Resende, 2011].

Segundo descrito em SNDlib, parte do conjunto de instâncias da biblioteca foi obtido através de parcerias entre instituições de ensino, fabricantes de equipamentos e operadoras de rede. Outra instâncias, por sua vez,são referências em grandes projetos de pesquisas na Europa, tendo, como participantes, grandes operadoras de telecomunicação e fabricantes de equipamentos. Outras instâncias foram utilizadas por outros autores em problemas de otimização e disponibilizadas para a biblioteca SNDlib e algumas foram alteradas através de, por exemplo, aumento do número de links, para torná-las mais desafiadoras. A matriz de tráfego de cada instância é assimétrica, isto é, pode existir uma demanda de um nó i para um nó j e não existir uma demanda no caminho contrário.

O valor definido para um braço ROADM foi $ROADM_{COST} = 500000$; o valor para um OA foi $OA_{COST} = 250000$; o valor para um OT foi $OT_{COST} = 50000$, o valor para a distância na fibra ótica em que deve ser instalado um OA foi $L_{OA} = 100$, o valor para a

distância na fibra ótica em que deve ser instalado um OT foi $L_{OT} = 500$ e o valor para a quantidade de comprimentos de onda suportados por cada fibra ótica foi $\mu = 100$.

A heurística IGS-PIFRO-Paralelo foi codificada utilizando a linguagem de programação C++ e compilada com o compilador TDM-GCC versão 5.1.0. Para a implementação das técnicas de paralelização citadas, foi utilizada a biblioteca OPENMP, apresentada em [Dagum and Menon, 1998]. Os experimentos foram executados em um computador com processador Intel Core I5, tendo 2.76 GHZ, com 4 GB de memória RAM e 4 cores físicos.

Dado que em [dos Reis et al., 2013] já foi demonstrado que até o momento a melhor heurística para PIFRO disponível na literatura é IGS-PIFRO, a qualidade de suas soluções foi descartada, sendo exibido neste artigo apenas o *Speedup* proporcionado pela paralelização desta nas figuras 5, 6 e 7. Nestas figuras, suas legendas exibem respectivamente o nome da heurística construtiva que foi utilizada por IGS-PIFRO-Paralelo, a configuração de seus modos de perturbação (Mono para monoperturbação e Mult para multiperturbação) e a configuração sobre a sincronização de soluções melhoradas pelas *threads* (Sinc). O sinal " + " anterior ao parâmetro indica que o mesmo estava ativo no momento a execução, enquanto o sinal −" antecedendo o parâmetro indicado que o mesmo estava desativado. O tempo de execução foi medido através da soma em segundos demandada para cada configuração atingir um valor alvo comum a todas a elas.

Conforme pode-se observar, estes gráficos/experimentos corroboram com o que é explicado na Seção 2, onde detalha-se a possibilidade de ser atingido um *Speedup* acima ou abaixo do esperado por heurísticas/metaheurísticas.

7 CONCLUSÕES

Neste trabalho foi avaliado o desempenho da heurística IGS-PIFRO-Paralelo utilizando técnicas de paralelismo. Verificando o *Speedup* alcançado pelas diferentes configurações da heurística, nota-se que sempre alguma configuração permite que ela tenha um *Speedup* superlinear para todas as instâncias. Para trabalhos futuros, pretende-se determinar quais as configurações que são melhores para cada conjunto de instâncias com características específicas.

REFERÊNCIAS

início IGS(*qtthreads*, *sincronizar*, *multiperturbacao*)
1. **faça em paralelo** de acordo com qtthreads
2. $s_{thread} \leftarrow$ Construção();
3. $s^{'}_{thread} \leftarrow$ BuscaLocal(s_{thread});
4. **fim-faça-em-paralelo**
5. s^{*}_{thread}, s_{thread}, $s^{'}_{thread} \leftarrow$ MelhorSolução($s^{'}_{thread}$);
6. *Definir início de sessão crítica*
7. **se** s^{*}_{thread} for melhor que s^{*}_{global} **então**
8. $s^{*}_{global} \leftarrow s^{*}_{thread}$;
9. **fim-se**
10. *Definir fim da sessão crítica*
11. **enquanto** critério de parada não for satisfeito **faça**
12. **faça em paralelo** de acordo com qtthreads
13. **se** multiperturbacao = verdadeiro **então**
14. $s^{'}_{thread} \leftarrow$ Perturbação(s_{thread}, (ValorDaPerturbação));
15. **senão**
16. $s^{'}_{thread} \leftarrow$ Perturbação(s_{thread});
17. **fim-se**
18. $s^{'}_{thread} \leftarrow$ BuscaLocal($s^{'}_{thread}$);
19. **se** $s^{'}_{thread}$ for melhor que s^{*}_{thread} **então**
20. $s^{*}_{thread} \leftarrow s^{'}_{thread}$;
21. **se** sincronizar = verdadeiro **então**
22. *Definir início de sessão crítica*
23. **se** s^{*}_{thread} for melhor que s^{*}_{global} **então**
24. $s^{*}_{global} \leftarrow s^{*}_{thread}$;
25. **fim-se**
26. *Definir fim da sessão crítica*
27. *Impor barreira de sincronização*
28. $s^{*}_{thread} \leftarrow s^{*}_{global}$;
29. **fim-se**
30. **fim-se**
31. **se** CritérioDeAceitação (s_{thread}, $s^{'}_{thread}$, s^{*}_{global}) **então**
32. $s_{thread} \leftarrow s^{'}_{thread}$;
33. **fim-se**
34. **fim-faça-em-paralelo**
35. **fim-enquanto**
36. **retorne** s^{*}_{global}

fim

Figura 2: Pseudocódigo da metaheurística IGS-PARALELO.

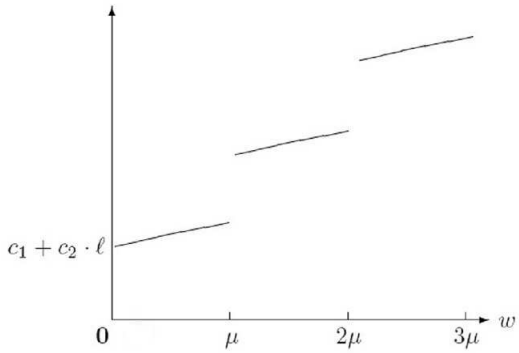

Figura 3: Representação gráfica da função $F_{WDM}(w, l)$..

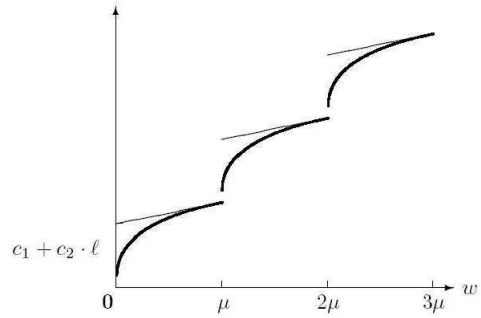

Figura 4: Representação gráfica de $F'_{WDM}(w, l, p, P)$ para $p = 1$ e $P = 5$.

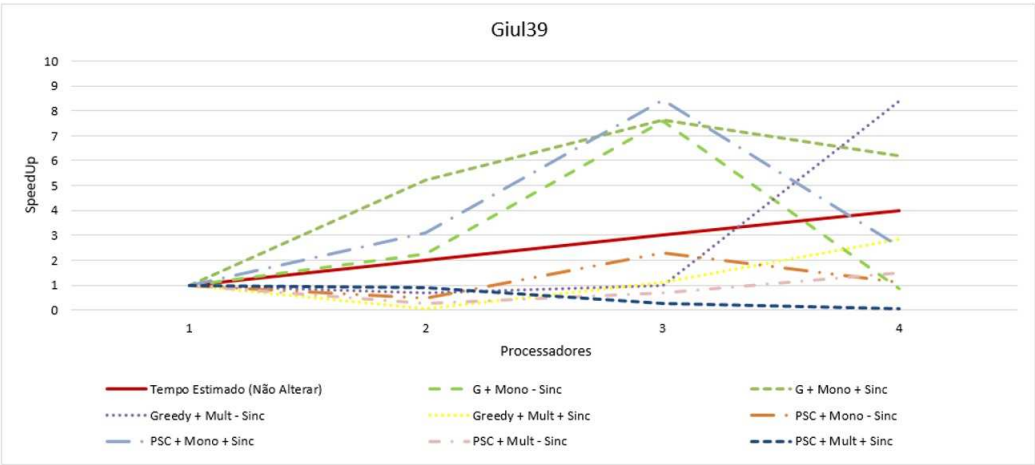

Figura 5: *Speedup* de IGS-PIFRO-Paralelo na instância GIUL39.

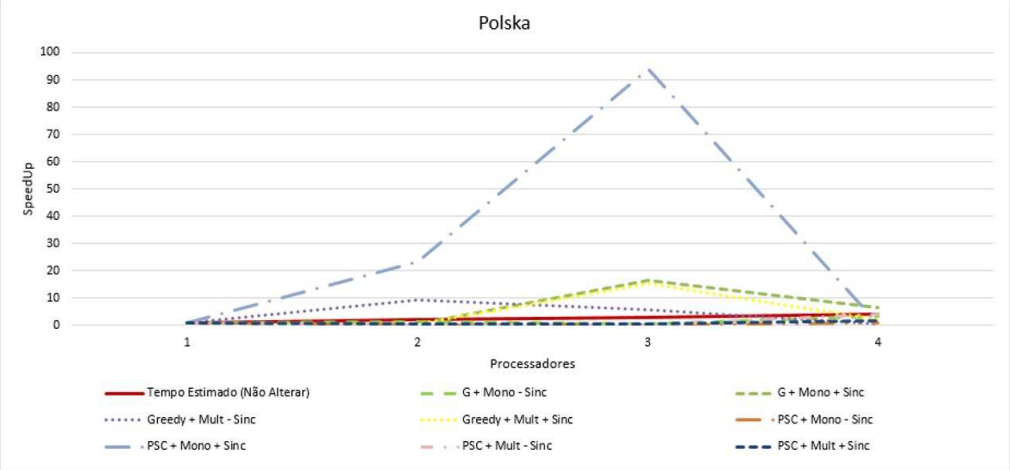

Figura 6: *Speedup* de IGS-PIFRO-Paralelo na instância POLSKA.

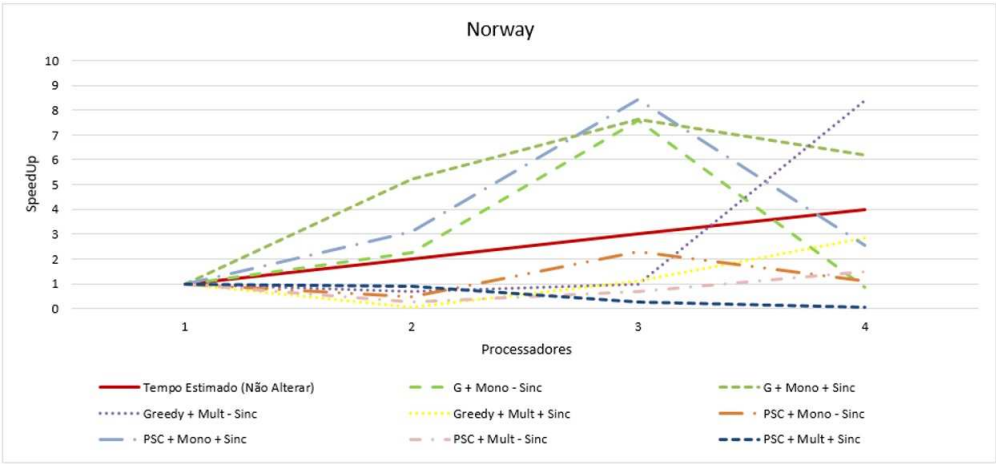

Figura 7: *Speedup* de IGS-PIFRO-Paralelo na instância NORWAY.

Referências Bibliográficas

[Alba, 2005] Alba, E. (2005). *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, NJ, USA.

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM.

[Antonakopoulos and Zhang, 2007] Antonakopoulos, S. and Zhang, L. (2007). Heuristics for fiber installation in optical network optimization. In *Proceedings of the 2007 IEEE Global Telecommunications Conference*, Washington.

[Bader et al., 2002] Bader, D. A., Moret, B. M. E., and Sanders, P. (2002). Algorithm engineering for parallel computation. In Fleischer, R., Moret, B., and Schmidt, E. M., editors, *Experimental Algorithmics: From Algorithm Design to Robust and Efficient Software*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Crainic et al., 2014] Crainic, T. G., Davidović, T., and Ramljak, D. (2014). Designing parallel meta-heuristic methods. In Belic, A. and Milutinovic, V. Despotovic-Zrakic, M. I., editors, *High Performance and Cloud Computing in Scientific Research and Education,*, pages 260–280. IGI Global,.

[Dagum and Menon, 1998] Dagum, L. and Menon, R. (1998). Openmp: An industry-standard api for shared-memory programming. *IEEE Comput. Sci. Eng.*, 5(1):46–55.

[dos Reis et al., 2013] dos Reis, D. M., Noronha, T. F., and de Souza, S. R. (2013). Iterated local search for fiber installation in optical network optimization. *Electronic Notes in Discrete Mathematics*, 41:277 – 284.

[Fanjul-Peyroa and Ruiz, 2010] Fanjul-Peyroa, L. and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207:55–69.

[Framinan and Leisten, 2008] Framinan, J. and Leisten, R. (2008). A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum*, 30:787–804.

[Gagnaire and Doumith, 2007] Gagnaire, M. and Doumith, E. (2007). An iterative greedy algorithm for scheduled traffic grooming in WDM optical networks. In *Advanced Networks and Telecommunication Systems First International Symposium On*.

[Garcia et al., 1998] Garcia, B. L., Mahey, P., and LeBlanc, L. J. (1998). Iterative improvement methods for a multiperiod network design. *European Journal of Operational*

Research, 110:150–165.

[Gonçalves and Resende, 2011] Gonçalves, J. F. and Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487–525.

[Goulart et al., 2011] Goulart, N., Dias, L. G. S., de Souza, S. R., and Noronha, T. F. (2011). Biased random-key genetic algorithm for fiber installation in optical network optimization. In *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*, pages 2267–2271.

[Hertz and de Werra, 1987] Hertz, A. and de Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, 39:345–351.

[Hill and Marty, 2008a] Hill, M. D. and Marty, M. R. (2008a). Amdahl's law in the multicore era. *Computer*, 41(7):33–38.

[Hill and Marty, 2008b] Hill, M. D. and Marty, M. R. (2008b). Amdahl's law in the multicore era. *Computer*, 41(7):33–38.

[Hyytiä and Virtamo, 1998] Hyytiä, E. and Virtamo, J. (1998). Wavelength assignment and routing in WDM networks. In *Fourteenth Nordic Teletraffic Seminar*, pages 31–40, Copenhagen.

[Lourenço et al., 2010] Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 363–397. Springer.

[Noronha and Ribeiro, 2006] Noronha, T. F. and Ribeiro, C. C. (2006). Routing and wavelength assignment by partition coloring. *European Journal of Operational Research*, 171:797–810.

[Pacheco, 2011] Pacheco, P. (2011). *An Introduction to Parallel Programming*. Morgan Kaufmann Publishers, Inc.

[Reis et al., 2012] Reis, D. M., Fuscaldi, F., Souza, S., and Noronha, T. F. (2012). Heurísticas para o problema de instalação de fibras em redes Óticas. In *Anais do SPOLM 2012 – Simpósio de Pesquisa Operacional da Marinha*.

[Ruiz and Stützle, 2007] Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *uropean Journal of Operational Research*, 177:2033–2049.

[Ruiz and Stützle, 2008] Ruiz, R. and Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187:1143–1159.

[Sun and Chen, 2010] Sun, X.-H. and Chen, Y. (2010). Reevaluating amdahl's law in the multicore era. *Journal of Parallel and Distributed Computing*, 70(2):183–188.

[Tullsen et al., 1995] Tullsen, D. M., Eggers, S. J., and Levy, H. M. (1995). Simultaneous multithreading: Maximizing on-chip parallelism. In *Proceedings 22nd Annual International Symposium on Computer Architecture*, pages 392–403.

[Zuse-Institute,] Zuse-Institute. Survivable fixed telecommunication network design.

`http://sndlib.zib.de/home.action`, Visitado pela ï¿$\frac{1}{2}$ltima vez em 26 de junho, 2013.