LI Simpósio Brasileiro de Pesquisa Operacional
Limeira–SP, 02 a 06 de setembro de 2019

APLICAÇÃO DA METODOLOGIA GERAR-E-RESOLVER AO PROBLEMA DA ARBORESCÊNCIA MÍNIMA DE COBERTURA

Jean Lucas de Souza Toniolli
Concordia University, Computer Science and Software Engineering, Master of Computer Science
1515 St. Catherine W., EV 3.150, Montreal, Quebec, Canada, H3G 2W1
toniolli@encs.concordia.ca

Iago do Nascimento Almeida, Gabriel Lopes Maia, Napoleão Nepomuceno
Universidade de Fortaleza, Programa de Pós-Graduação em Informática Aplicada
Av. Washigton Soares 1321, CEP: 60811-905, Fortaleza, Ceará, Brasil
{ialmeida,glmaia}@edu.unifor.br, napoleaovn@unifor.br

RESUMO

Neste trabalho, é avaliado o uso da metodologia Gerar-e-Resolver para o Problema da Arborescência Mínima de Cobertura, utilizando as metaheurísticas Têmpera Simulada e Algoritmo Genético. O Gerar-e-Resolver é uma metodologia híbrida para lidar com problemas difíceis de otimização combinatória, reduzindo artificialmente o espaço de busca de soluções. Essa abordagem tem sido bastante eficaz para problemas de corte e empacotamento e problemas de redes sem fio. Experimentos computacionais foram realizados em um conjunto variado de instâncias para comparar as metaheurísticas e o resolvedor exato. Os resultados obtidos mostram que o Gerar-e-Resolver é competitivo com o resolvedor exato para instâncias de tamanho pequeno a médio, e também é demonstrado que a metodologia supera significativamente o resolvedor exato para instâncias de tamanhos maiores, evidenciando a eficácia da abordagem. Além disso, é analisado que o desempenho do Algoritmo Genético cai drasticamente em relação à Têmpera Simulada conforme o tamanho e densidade das instâncias aumentam.

PALAVRAS CHAVE. Metaheurísticas. Métodos Híbridos. Arborescência Mínima de Cobertura.

Tópicos: MH – Meta-heurísticas. OC – Otimização Combinatória.

ABSTRACT

In this work, we evaluate the use of the Generate-and-Solve methodology to tackle the Minimum Covering Arborescence Problem using Simulated Annealing and Genetic Algorithm metaheuristics. Generate-and-Solve is a hybrid methodology for dealing with difficult combinatorial optimization problems, artificially reducing the search space for solutions. This approach has been quite effective to deal with cutting and packing problems and wiresless network problems. Computational experiments were performed on a varied set of instances to compare the metaheuristics and the exact solver. The results show that Generate-and-Solve is competitive with the exact solver for small to medium size instances, and it is also demonstrated that the methodology significantly outperforms the exact solver for instances of larger sizes, evidencing the effectiveness of the approach. In addition, we analyze that the performance of the Genetic Algorithm falls dramatically in relation to the Simulated Annealing as the size and density of the instances increases.

KEYWORDS. Metaheuristics. Hybrid Methods. Minimum Covering Arborescence.

Paper Topics: MH – Metaheuristics. OC – Combinatorial Optimization.

https://proceedings.science/p/106895?lang=pt-br

1. Introdução

Otimização Combinatória (OC) é uma área da ciência computacional teórica e da matemática aplicada que trata dos problemas nos quais uma solução ideal deve ser encontrada dentre um largo conjunto finito de soluções, que podem ser discretas ou transformadas em discretas [Cook et al., 1997]. Vários desses problemas são NP-difíceis, ou seja, não se conhece algoritmo determinístico para resolvê-los em tempo polinomial, o que, na maioria das vezes, restringe a aplicação de métodos exatos para achar a solução de instâncias do mundo real.

Devido à grande dificuldade e à enorme relevância prática de se resolver problemas complexos do mundo real, no campo de OC foram realizados esforços de pesquisa para o desenvolvimento de métodos heurísticos com o objetivo de encontrar boas soluções aproximadas. Um dos frutos de tal esforço foi a caracterização de metaheurísticas, termo introduzido inicialmente por [Glover, 1986], que essencialmente se refere a uma ampla classe de estratégias para otimização aproximada e resolução de problemas que são independentes do problema. Tais metaheurísticas provaram ser altamente úteis para resolver problemas de OC de forma aproximada.

No entanto, há um consenso geral de que ainda existe espaço para a conceitualização de soluções mais sofisticadas nesta área de pesquisa. Nesse contexto, é cada vez mais evidente a quantidade de algoritmos apresentados que não seguem puramente os conceitos de uma única metaheurística tradicional, ou seja, combinações de conceitos decorrentes de diferentes metaheurísticas com outras técnicas de Pesquisa Operacional (PO). De acordo com [Blesa et al., 2005], essas combinações de métodos exatos mais convencionais e diferentes metaheurísticas em uma única estrutura de otimização são referidas como Metaheurísticas Híbridas (MHs).

Seguindo esta metodologia, que cumpre o preceito de hibridização mencionado acima, foi introduzido o *framework* Gerar-e-Resolver (G&R) por [Nepomuceno, 2006], um método híbrido que utiliza uma metaheurística para gerar instâncias reduzidas de um problema que são passadas para um resolvedor exato. Tendo em mente o objetivo de aproximação da solução ótima do problema original, estas soluções reduzidas servem como *feedback* para a metaheurística escolhida guiar a geração de novas instâncias reduzidas.

Este trabalho tem como objetivo investigar a performance do G&R, utilizando diferentes metaheurísticas – Têmpera Simulada (TS) e Algoritmos Genéticos (AG) – como geradores de instâncias reduzidas para o Problema da Arborescência Mínima de Cobertura (PAMC), que é uma extensão do problema atacado em [Blum e Calvo, 2015], e comparar a sua eficiência em relação a um resolvedor exato, além de realizar a comparação de performance de cada metaheurística.

O restante do artigo está estruturado da seguinte forma: na Seção 2, faz-se um levantamento bibliográfico sobre metodologias que utilizam MHs, explorando mais em detalhes o *framework* G&R. Na Seção 3, apresenta-se detalhadamente o modelo matemático proposto para o problema investigado neste trabalho. Na Seção 4, explica-se detalhadamente como foram implementadas as metaheurísticas para o resolvedor de instâncias reduzidas. A Seção 5 tem como propósito mostrar os resultados computacionais obtidos e realizar comparações entre os resultados das metaheurísticas testadas e o resolvedor exato para o problema atacado neste trabalho. Na Seção 6, as conclusões e sugestões de trabalhos futuros são apresentadas.

2. Trabalhos Relacionados

MHs é um campo relativamente recente de pesquisa [Blum e Raidl, 2016], no qual a combinação de ideias de métodos diversos de otimização deu origem a diferentes esquemas de hibridização [Talbi, 2002; Raidl, 2006]. De acordo com [Puchinger e Raidl, 2005], muitos algoritmos híbridos buscam gerar soluções ótimas no menor tempo possível, enquanto outros tentam principalmente melhorar soluções heurísticas. [Blum e Raidl, 2016] ressaltam, entretanto, que não

https://proceedings.science/p/106895?lang=pt-br

é trivial generalizar uma hibridização, pois um híbrido pode funcionar melhor para certos problemas e ter um desempenho ruim para outros.

MHs que combinam técnicas de programação matemática com metaheurísticas têm recebido muita atenção como uma alternativa que busca tanto eficiência na utilização dos recursos computacionais quanto qualidade dos resultados alcançados [Blum et al., 2011]. A principal motivação deste tipo de hibridização é explorar o caráter complementar destas diferentes estratégias de otimização.

O G&R faz parte desta categorização de métodos híbridos e aproveita-se do fato de que resolvedores genéricos de Programação Linear Inteira (PLI), como o CPLEX [Cplex, 2017] e o GUROBI [Gurobi, 2018], geralmente são efetivos em termos de tempo até um certo tamanho de problema. Quando o problema é muito grande para ser resolvido diretamente por um resolvedor de PLI, existem formas eficazes de se obter instâncias reduzidas do problema que ainda contenham soluções de alta qualidade, ou até mesmo ótimas, para o problema original. A instância reduzida deverá, entretanto, ser pequena o suficiente para que o resolvedor PLI seja capaz de executar em tempo aceitável.

Os componentes conceituais do G&R, o Gerador de Instâncias Reduzidas (GIR) e o Resolvedor de Instâncias Reduzidas (RIR), são ilustrados na Figura 1. O GIR representa uma metaheurística que é responsável por gerar sub-instâncias do problema original, de tal forma que elas ainda preservem as restrições do problema original, ou seja, uma solução da sub-instância também será uma solução viável para o problema original. O RIR encapsula um método exato encarregado de resolver as instâncias reduzidas geradas pelo GIR. Além disso, o RIR retorna informações referentes à solução ótima de cada sub-instância para serem usadas na definição de sua qualidade e guiar a busca da metaheurística. Finalmente, a solução para o problema original será a melhor solução obtida ao longo da execução da metaheurística.

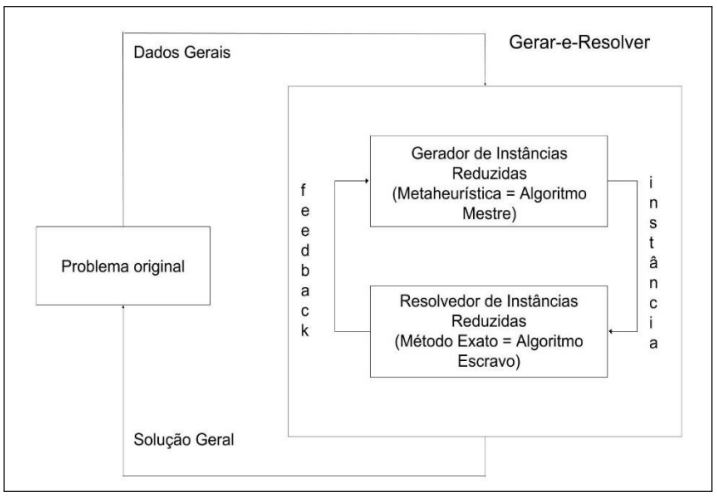


Figura 1: *Framework* Gerar-e-Resolver

O G&R foi empregado em sua maioria na resolução de problemas de corte e empacotamento, alcançando resultados de alta qualidade para esses problemas [Nepomuceno et al., 2007b,a, 2008]. O G&R foi utilizado ainda para na resolução de problemas em redes sem fio [Coudert et al., 2010, 2011] e foi revisitado em [Saraiva et al., 2013; Coudert et al., 2010] para empregar a metaheurística TS como GIR.

Além do G&R, a ideia do uso de técnicas de redução de instâncias foi explorada em trabalhos anteriores de diversas maneiras na literatura [Clements et al., 1997; Massen et al., 2012, 2013; Applegate et al., 1999; Cook e Seymour, 2003]. Recentemente, [Blum et al., 2016] introduziram o método Construct, Merge, Solve, and Adapt (CMSA), que pode ser considerada uma instanciação do *framework* G&R. Nesse algoritmo, o GIR gera subinstâncias através da junção de soluções obtidas de maneira probabilística por uma heurística específica para o (ou dependente do) problema em consideração. Também em [Blum et al., 2016], um mecanismo de envelhecimento de componentes de uma solução foi introduzido para controlar o crescimento das subinstâncias e, consequentemente, limitar o tempo necessário para resolvê-las. Neste trabalho, o mecanismo de envelhecimento é empregado no G&R.

3. O Problema da Arborescência Mínima de Cobertura

O PAMC é um problema NP-Difícil [Bang-Jensen e Gutin, 2008], que pertence à família dos problemas de Arborescência Enraizada de Peso Mínimo [Rao e Sridharan, 1992]. Nestes problemas, é dado um Grafo Direcionado Acíclico (GDA) com raiz em um nó predefinido e com pesos inteiros nos arcos. Um subgrafo do problema que estabelece uma arborescência dirigida – na qual todos os arcos apontam na direção oposta à raiz – será uma solução, mas não necessariamente de cobertura. O objetivo é achar uma solução válida com peso mínimo, onde o custo é dado pela soma dos pesos dos arcos da solução.

Vale ressaltar que esses problemas tem uma grande importância prática ligada a várias aplicações do mundo real, em áreas como visão computacional e planejamento de produção em múltiplos estágios. Como exemplos, temos o problema da reconstrução automatizada de estruturas de árvores consistentes em sistemas biológicos (por exemplo, redes dendríticas e vasculares) a partir de imagens ruidosas [Türetken et al., 2011] e a aplicação de extração de minério, descrita por [Philpott e Wormald, 2000], onde o PAMC modela a situação em que se deseja encontrar a melhor política para explorar blocos de uma mina.

O problema estudado neste trabalho é bastante relacionado com o problema investigado em [Duhamel et al., 2008], uma variante mais geral do problema. Considera-se aqui, entretanto, que o grafo de entrada possui um nó raiz designado. Dado um GDA denotado por $G = (V, A)$, em que $V = \{v_1, ..., v_n\}$ é o conjunto de n vértices e A é o conjunto de m arcos tal que $A \subseteq \{(i, j)|i \neq j \in V\}$. Cada arco $a \in A$ tem um peso inteiro $w_a \in \mathbb{Z}$. É também definido um conjunto $X \subseteq V$ de vértices que devem fazer parte da solução. Soluções válidas são definidas como qualquer arborescência $T = (V(T), A(T))$, tal que $V(T) \subseteq V$ e $A(T) \subseteq A$, enraizada no vértice $v_1 \in V$, que contenha todos os nós requeridos, ou seja, $X \subseteq V(T)$. O peso de uma arborescência é definido como a soma dos pesos de todos os seus arcos:

$$f(T) = \sum_{a \in A(T)} w_a$$

O objetivo do PAMC é encontrar uma arborescência T' tal que o valor de $f(T')$ seja menor ou igual ao valor do peso de qualquer outra arborescência válida.

Um exemplo do PAMC é mostrado na Figura 2, onde (a) mostra um grafo de entrada com oito vértices e quinze arcos e (b) mostra a solução ótima com valor -17. Os vértices preenchidos na cor preta formam o conjunto X, isto é, eles devem ser incluídos em qualquer solução válida.

Um modelo em PLI para o problema, proposto em [Blum et al., 2016], trabalha com um grafo expandido, adicionando um vértice v_0 e um arco de peso zero que o conecta à raiz original do grafo. Nesta formulação, o conjunto de arcos predecessores de um arco a é definido como $pred(a)$, e o conjunto de arcos que apontam para um vértice v é definido como $in(v)$. Para cada arco a,

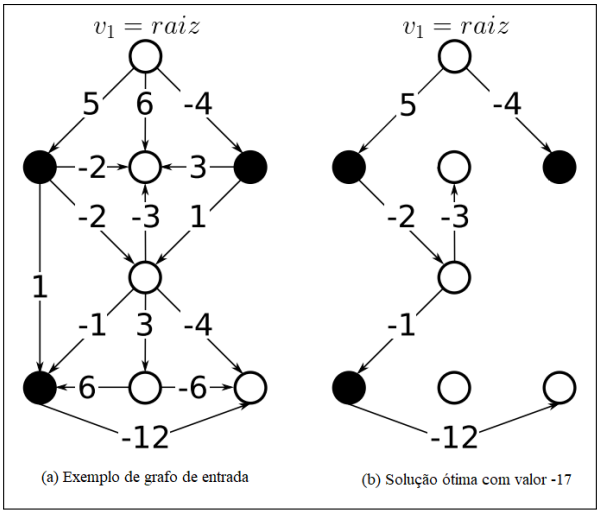


Figura 2: Instância do PAMC [Blum et al., 2016]

associa-se uma variável de decisão binária x_a que indica se a faz (valor 1) ou não (valor 0) parte da solução. O modelo é definido a seguir.

$$\min \sum_{a \in A} w_a x_a \tag{1}$$

$$\text{s.t.} \quad x_{v_0,v_1} = 1 \tag{2}$$

$$\sum_{a \in in(v_i)} x_a \leq 1 \qquad ,\forall v_i \in V \setminus X \tag{3}$$

$$\sum_{a \in in(v_i)} x_a = 1 \qquad ,\forall v_i \in X \tag{4}$$

$$x_a - \sum_{a' \in pred(a)} x_{a'} \leq 0 \qquad ,\forall a \in A \tag{5}$$

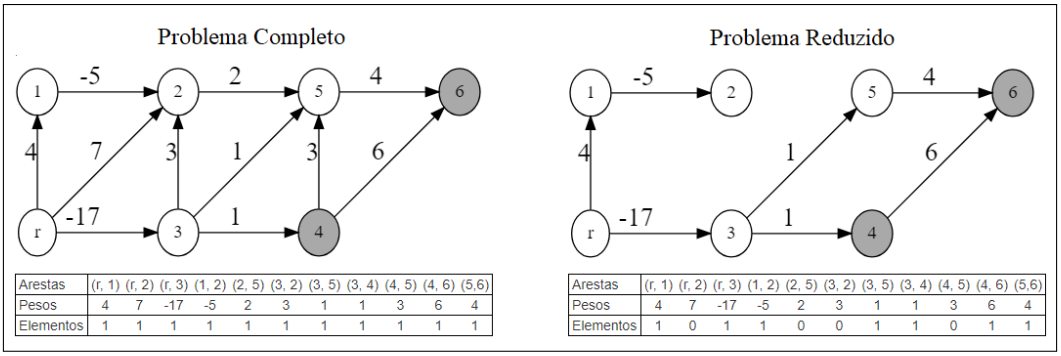
$$x_a \in \{0,1\} \qquad ,\forall a \in A \cup \{(v_0,v_1)\} \tag{6}$$

A função objetivo (1) busca minimizar o peso da arborescência. A restrição (2) assegura que o arco artificial (v_0, v_1) deve fazer parte da solução. Em (3), assegura-se que todos os vértices não obrigatórios tenham, no máximo, um único arco apontando para si formando a arborescência. A restrição (4) garante que todos os vértices em X são obrigados a fazer parte da solução. Em (5), define-se que qualquer arco que faz parte da solução deve ter seu vértice predecessor alcançável na arborescência. A restrição (6) expressa o domínio das variáveis de decisão.

4. Aplicação da Metodologia Gerar-e-Resolver

A formulação em PLI do PAMC contém $O(m)$ variáveis de decisão. As instâncias reduzidas do problema podem ser geradas ao considerar apenas um subconjunto de variáveis de decisão do problema original, conforme ilustrado na Figura 3.

A representação é feita através de um vetor binário, onde cada elemento representa uma variável de decisão (um arco do grafo). Um valor 0 (zero) significa que determinada variável de decisão não faz parte da subinstância. Por outro lado, um valor 1 (um) significa que a variável de



Arestas	(r, 1)	(r, 2)	(r, 3)	(1, 2)	(2, 5)	(3, 2)	(3, 5)	(3, 4)	(4, 5)	(4, 6)	(5,6)
Pesos	4	7	-17	-5	2	3	1	1	3	6	4
Elementos	1	1	1	1	1	1	1	1	1	1	1

Arestas	(r, 1)	(r, 2)	(r, 3)	(1, 2)	(2, 5)	(3, 2)	(3, 5)	(3, 4)	(4, 5)	(4, 6)	(5,6)
Pesos	4	7	-17	-5	2	3	1	1	3	6	4
Elementos	1	0	1	1	0	0	1	1	0	1	1

Figura 3: Redução de uma instância do PAMC

decisão faz parte da subinstância. Portanto, a subinstância gerada é um modelo em PLI contendo todas as restrições presentes na formulação do problema, mas apenas um subconjunto dos arcos do grafo. Tendo em vista a restrição (4), pela qual os vértices do conjunto X são obrigados a fazer parte da solução, um Gerador Probabilístico de Soluções (GPS) [Blum et al., 2016] foi utilizado para se obter uma solução inicial e, a partir daí, garantir a existência de soluções viáveis em todas as subinstâncias submetidas ao RIR, visto que as arcos utilizadas na solução ótima de uma subinstância corrente são mantidas na geração de subinstâncias vizinhas.

Para o GIR, fez-se uso das metaheurísticas AG e TS. Cada subinstância gerada deve ser resolvida pelo RIR, cujo valor da função objetivo – que representa a soma dos pesos dos arcos utilizados na arborescência – é assumido como valor de aptidão da solução para comparação com outras soluções, guiando o processo de geração de subinstâncias. Além disso, um mecanismo de envelhecimento age adicionalmente na geração das subinstâncias pela metaheurística. Quando arcos presentes em uma subinstância repetidamente não fazem parte de sua solução ótima, são então removidos da mesma para enxugar seu tamanho, acelerando sua resolução pelo resolvedor exato. Arcos adicionados na subinstância têm sua idade definida como 0. Após a execução do RIR, arcos da subinstância que não foram usados na solução ótima envelhecem através de um incremento de 1, enquanto a idade daqueles que foram efetivamente utilizados volta a ser 0. Arcos que atingem a idade máxima – um parâmetro do GIR – são removidos da sub-instância. O critério de parada para ambas metaheurísticas implementadas é um tempo limite de execução.

4.1. Têmpera Simulada

O algoritmo TS inicia com uma alta temperatura T e começa a busca a partir de uma configuração inicial S_0, obtida a partir de uma execução do GPS. A configuração inicial é resolvida pelo RIR e os arcos utilizados na solução ótima da subinstância são fixados na configuração.

O laço de iterações, que caracteriza o procedimento principal, gera aleatoriamente, em cada iteração, um único vizinho S_i a partir da configuração S_{i-1}. Essa pertubação aleatória é baseada em uma taxa de perturbação P_r, onde cada elemento da configuração poderá ser invertido de acordo com P_r. Após a pertubação dessa nova configuração, esta é passada para o RIR que retorna um valor de energia para a mesma. De acordo com o valor de energia desse vizinho e a temperatura atual, o algoritmo decidirá se ele será aceito como configuração atual ou se será descartado. A aceitação dessa nova configuração é baseada na descrição original da TS [Kirkpatrick et al., 1983], que permite a aceitação de configurações que não melhoram o valor de energia de acordo com a

seguinte função de probabilidade:

$$PA(e, e', T) = \begin{cases} 1 & \text{se } e' < e \\ exp(\frac{e - e'}{T}) & \text{caso contrário} \end{cases} \quad (7)$$

onde e é a energia da configuração atual, e' a energia da configuração vizinha e T a temperatura. Além disso, quando a configuração vizinha for melhor esta será sempre aceita.

Caso este vizinho S_i seja aceito, os arcos utilizados em sua solução são fixados para garantir a viabilidade de qualquer subinstância gerada a partir dele. Ao fim da iteração, a temperatura T é resfriada de acordo com uma taxa de resfriamento C_r e o laço continua. Se a metaheurística chega à temperatura de resfriamento menor que 1, ela reinicia tomando a solução encontrada como nova configuração inicial.

4.2. Algoritmo Genético

O AG evolui uma população de P cromossomos inicialmente gerados a partir de P soluções obtidas com o GPS. Esses cromossomos são resolvidos pelo RIR e, então, os arcos que foram usados na solução ótima serão fixados no cromossomo e em seus descendentes para a garantia de viabilidade. Uma nova geração é constituída, sendo P_e cromossomos-pais selecionados por elitismo e $P - P_e$ cromossomos-filhos originados por intermédio dos operadores de seleção, cruzamento e mutação. Após a aplicação desses operadores, o RIR resolve cada cromossomo, gerando um valor de aptidão (valor da função objetivo) para cada um deles.

Seleção

O operador de seleção implementado é a Seleção por Roleta [Holland, 1992]. O princípio da seleção de roleta é uma pesquisa linear através de uma roleta com segmentos ponderados proporcionalmente aos valores de aptidão do cromossomo – quanto melhor, maior o segmento. Então, a roleta virtual é girada, e o cromossomo correspondente ao segmento no qual a roleta para é selecionado. O processo é repetido até se obter o número desejado de cromossomos.

Cruzamento

O operador de cruzamento implementado é o Cruzamento de Ponto Único [Holland, 1975]. Neste cruzamento, um único ponto em ambos os cromossomos-pais é selecionado dentro do intervalo discreto $\{0, 1, \ldots, l_c - 1\}$, onde l_c é o número de elementos do cromossomo (ou seja, o número de arcos do grafo). Os cromossomos-pais são divididos no ponto de cruzamento, e os dados além desse ponto são trocados entre os cromossomos-pais. Este cruzamento será aplicado com uma dada probabilidade, determinada pela taxa de cruzamento T_c. Os arcos fixados de um cromossomo-pai, entretanto, são carregados para um cromossomo-filho para garantir a existência de uma solução viável na subinstância.

Mutação

A mutação pode ser definida como um pequeno ajuste aleatório no cromossomo, para obter uma nova solução. Este operador é usado para manter e introduzir diversidade na população genética. Após a operação de cruzamento, o operador de mutação é aplicado em cada elemento dos dois cromossomos-filhos, de acordo com uma taxa de mutação M_r, invertendo os valores dos elementos. Note-se que, se o elemento estiver fixado por conta da utilização do arco na solução ótima da subinstância, a mutação não será aplicada àquele elemento.

5. Resultados Computacionais

Utilizou-se o IBM ILOG CPLEX 12.8 [Cplex, 2017] para solução de modelos em PLI. As metaheurísticas do GIR foram desenvolvidas na linguagem Java e seus parâmetros foram calibrados

utilizando o software *Irace* [López-Ibánez et al., 2011], que é uma ferramenta de configuração automática de algoritmos probabilísticos. Os testes experimentais e a calibragem dos parâmetros foram executados em máquinas *Intel Pentium i7 7700 3.60 GHz 8 GB RAM*. Os seguintes algoritmos foram considerados para o experimento:

- $PLI_{completo}$: a aplicação do otimizador CPLEX para o modelo completo de cada instância considerada do PAMC;
- $G\&R_{ts}$: o *framework* G&R com o algoritmo Têmpera Simulada como Gerador de Instâncias Reduzidas;
- $G\&R_{ag}$: o *framework* G&R com o Algoritmo Genético como Gerador de Instâncias Reduzidas.

Um conjunto de instâncias foi gerado pelo método descrito por [Blum et al., 2016]. A quantidade de nós do grafo $G = (V, A)$ é denotada por $N \in \{500, 1000, 5000\}$. A probabilidade $P_{arc} \in \{0,1; 0,3; 0,5\}$ é usada para determinar se cada possível arco a apontando de um vértice $v_i \in V$ para outro vértice $v_j \in V$ – onde $i < j$ – é adicionado ao conjunto de arcos A ou não. O parâmetro $perc \in \{1\%, 10\%, 20\%\}$ determina o percentual de nós que fazem parte do conjunto X. Gerando 10 instâncias do problema para cada combinação dos valores possíveis de N, P_{arc}, e $perc$, temos o total de 270 instâncias do problema para o experimento.

5.1. Calibragem dos Algoritmos

Neste trabalho, o *Irace* foi aplicado para calibrar o GIR separadamente para cada uma das 9 combinações de N e P_{arc}. O único parâmetro comum a ambas metaheurísticas é a idade máxima age_{max} do mecanismo de envelhecimento. Foram considerados os seguintes valores para o $age_{max} \in \{-1, 1, 3, 5, 10\}$, onde -1 significa que os componentes da solução nunca são removidos da subinstância. Além disso, os algoritmos possuem vários parâmetros próprios que devem ser calibrados. Para a TS, os seguintes valores foram testados para os parâmetros: $T \in \{100; 1000; 10000\}$, $C_r \in \{0,005; 0,01; 0,03\}$ e $P_r \in \{0,001; 0,005; 0,01; 0,05\}$. Já para o AG, os seguintes valores foram considerados: $P \in \{10; 15; 20; 50; 100\}$, $M_r \in \{0,005; 0,01; 0,05\}$ e $T_c \in \{0,6; 0,7; 0,8\}$.

As configurações produzidas pelo *Irace* para a TS e o AG são apresentadas na Tabela 1.

	TS				AG			
$(N; P_{arc})$	age_{max}	T	C_r	P_r	age_{max}	P	M_r	T_c
(500; 0,1)	10	100	0,01	0,05	10	100	0,05	0,7
(500; 0,3)	5	1000	0,005	0,05	5	50	0,05	0,7
(500; 0,5)	5	1000	0,01	0,05	3	50	0,05	0,6
(1000; 0,1)	5	1000	0,03	0,05	5	20	0,05	0,7
(1000; 0,3)	3	1000	0,01	0,05	3	20	0,05	0,7
(1000; 0,5)	1	10000	0,01	0,01	3	15	0,005	0,8
(5000; 0,1)	3	10000	0,01	0,01	3	15	0,01	0,8
(5000; 0,3)	1	1000	0,01	0,01	1	10	0,01	0,6
(5000; 0,5)	3	1000	0,03	0,005	1	10	0,005	0,7

Tabela 1: Valores dos parâmetros sugeridos pelo *Irace* para a TS e o AG

https://proceedings.science/p/106895?lang=pt-br

(a) Resultados para instâncias com $perc = 1\%$

		$PLI_{completo}$			GR_{ts}		GR_{ag}	
N	P_{arc}	**objetivo**	**tempo**	**gap**	**média**	**tempo**	**média**	**tempo**
500	0,1	-940,1	0,8	0,0	-847,6	126,7	-940,1	173,8
	0,3	-13.450,8	5,3	0,0	-13.450,8	195,0	-11.846,0	226,8
	0,5	-28.030,8	13,1	0,0	-27.347,8	241,9	-19.871,4	232,5
1000	0,1	-15.263,9	20,9	0,0	-14.109,6	366,8	-15.263,9	464,3
	0,3	-62.414,8	156,8	0,0	-62.282,9	490,9	-41.175,1	498,6
	0,5	-106.522,0	435,1	0,0	-56.109,2	479,9	-38.784,9	132,6
5000	0,1	n/d	n/d	n/d	-404.623,0	2.474,6	-316.611,2	2.337,3
	0,3	n/d	n/d	n/d	-785.629,0	1943,4	-602.185,5	2229,5
	0,5	n/d	n/d	n/d	-1.322.799,0	2.245,3	-812.104,0	2457,0

(b) Resultados para instâncias com $perc = 10\%$

		$PLI_{completo}$			GR_{ts}		GR_{ag}	
N	P_{arc}	**objetivo**	**tempo**	**gap**	**média**	**tempo**	**média**	**tempo**
500	0,1	5.648,3	0,7	0,0	5.686,1	98,9	5.648,3	178,0
	0,3	-11.338,3	4,7	0,0	-11.333,0	192,9	-11.014,5	225,6
	0,5	-26.982,4	12,6	0,0	-26.367,3	243,5	-24.914,7	223,0
1000	0,1	-9.115,1	20,4	0,0	-7.783,7	337,1	-9.089,7	462,1
	0,3	-61.065,8	196,1	0,0	-60.101,4	485,6	-48.070,3	499,3
	0,5	-105.633,0	476,0	0,0	-81.565,6	497,9	-69.837,5	497,5
5000	0,1	n/d	n/d	n/d	-453.885,9	2.479,5	-425.426,7	2.492,6
	0,3	n/d	n/d	n/d	-932.884,0	1.625,7	-735.572,0	1.846,5
	0,5	n/d	n/d	n/d	-1.363.577,0	1.054,3	-1.115.695,0	2499,8

(c) Resultados para instâncias com $perc = 20\%$

		$PLI_{completo}$			GR_{ts}		GR_{ag}	
N	P_{arc}	**objetivo**	**tempo**	**gap**	**média**	**tempo**	**média**	**tempo**
500	0.1	10.887,8	0,8	0,0	10.948,8	89,5	10.887,8	154,8
	0.3	-9.801,0	6,1	0,0	-9.738,2	186,5	-9.559,9	226,1
	0.5	-25.827,6	18,7	0,0	-25.338	239,3	24.075,1	221,4
1000	0.1	-4.228,4	22,7	0,0	-3.025,5	387,5	-4.210,9	472,5
	0.3	-59.319,6	176,8	0,0	-59.319,6	480,5	-48.517,5	493,3
	0.5	-104.945,8	477,1	0,0	-90.837,2	496,9	-74.026,3	495,7
5000	0,1	n/d	n/d	n/d	-498.106,0	2.479,9	-441.293,0	2.488,7
	0,3	n/d	n/d	n/d	-876.935,0	2.498,3	-744.232,2	2.115,4
	0,5	n/d	n/d	n/d	-1.328.217,0	1.005,0	-1.072.565,5	1.898,1

Tabela 2: Resultados para as instâncias do PAMC

5.2. Análise dos Resultados

Os resultados experimentais para o conjunto de instâncias são mostrados na Tabela 2. Cada linha da tabela fornece resultados médios de 5 execuções das metaheurísticas para cada uma das 10 instâncias do PAMC da combinação de N, P_{arc} e $perc$. Para a execução dos algoritmos, foi usado um limite de tempo de computação de $N/2$ segundos. O melhor resultado obtido dentre os três algoritmos é marcado com um fundo cinza. Resultados com no máximo $0,5\%$ de diferença do obtido pelo melhor algoritmo estão sublinhados.

Podemos analisar em relação à aplicação do CPLEX ao $PLI_{completo}$ que o tamanho dos grafos de entrada possuem uma forte influência na dificuldade do problema. É visto nas tabelas que o $PLI_{completo}$ foi capaz de resolver para otimalidade todos os problemas com $N \in \{500, 1000\}$, entretanto, não conseguiu obter solução viável dentro do tempo limite para $N = 5000$. A porcentagem de vértices obrigatórios em uma solução, $perc$, não mostrou influência sobre o $PLI_{completo}$, visto que os tempos continuam parecidos mesmo com o acréscimo do valor de $perc$.

Para grafos com $n \in \{500, 1000\}$, o G&R foi quase sempre capaz de encontrar soluções ótimas ou próximo disso, assim sendo, competitivo com o CPLEX. Para grafos com densidades maiores, o G&R é consideravelmente mais rápido e eficaz que o CPLEX, pois vemos que com $N = 5000$ o CPLEX não consegue chegar a uma solução viável. Analisando a performance das metaheurísticas aplicadas ao GIR, é visto que, para grafos de entrada com $N = 500$ e $P_{arc} = 0,1$, o AG atinge o ótimo para todos os valores de $perc$. Porém, conforme a densidade e tamanho dos grafos aumentam, podemos ver que a performance do AG cai drasticamente, sendo superado pela TS com uma boa margem.

6. Conclusão

Neste trabalho, fez-se a aplicação da metodologia Gerar-e-Resolver ao Problema da Arborescência Mínima de Cobertura. Experimentos computacionais realizados em um conjunto variado de instâncias mostram que a metodologia G&R tem eficiência satisfatória para instâncias de tamanho moderado, embora seja superada pelo solucionador exato CPLEX. Entretanto, para instâncias de grande porte, a metodologia G&R superou o solucionador exato, o qual não consegue sequer encontrar solução viável em tempo hábil. O G&R, portanto, consegue tratar instâncias que, de outra forma, não seriam tratáveis pelo solucionar exato.

Em relação à comparação das metaheurísticas empregadas como Gerador de Instâncias Reduzidas, percebe-se que o AG obtém os melhores resultados para as menores instâncias. Porém, conforme o tamanho das instâncias aumenta, o desempenho do AG cai drasticamente. A TS como Gerador de Instâncias Reduzidas supera o AG para as instâncias de grande porte, exatamente aquelas onde a metodologia G&R se mostrou mais interessante.

A investigação de outras metaheurísticas para o Gerador de Instâncias Reduzidas é algo a se considerar para aumentar o escopo do estudo da metodologia, assim como a investigação de outros problemas de otimização combinatória.

Referências

Applegate, D., Bixby, R., Chvátal, V., e Cook, W. (1999). Finding tours in the tsp. Technical report.

Bang-Jensen, J. e Gutin, G. Z. (2008). *Digraphs: theory, algorithms and applications*. Springer Science & Business Media.

Blesa, M. J., Blum, C., Roli, A., e Sampels, M. (2005). Hybrid metaheuristics. *HM*.

Blum, C. e Calvo, B. (2015). A matheuristic for the minimum weight rooted arborescence problem. *Journal of Heuristics*, 21(4):479–499.

Blum, C., Pinacho, P., López-Ibáñez, M., e Lozano, J. A. (2016). Construct, merge, solve & adapt a new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68: 75–88.

Blum, C., Puchinger, J., Raidl, G. R., e Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151.

Blum, C. e Raidl, G. R. (2016). *Hybrid Metaheuristics: Powerful Tools for Optimization*. Springer.

Clements, D., Crawford, J., Joslin, D., Nemhauser, G., Puttlitz, M., e Savelsbergh, M. (1997). Heuristic optimization: A hybrid ai/or approach. In *Proceedings of the Workshop on Industrial Constraint-Directed Scheduling*.

Cook, W. e Seymour, P. (2003). Tour merging via branch-decomposition. *INFORMS Journal on Computing*, 15(3):233–248.

Cook, W. J., Cunningham, W., Pulleyblank, W., e Schrijver, A. (1997). *Combinatorial Optimization*. Wiley-Interscience. ISBN 047155894X.

Coudert, D., Nepomuceno, N., e Rivano, H. (2010). Power-efficient radio configuration in fixed broadband wireless networks. *Computer Communications*, 33(8):898–906.

Coudert, D., Nepomuceno, N., e Tahiri, I. (2011). Energy saving in fixed wireless broadband networks. In *Network Optimization*, p. 484–489. Springer.

Cplex, I. I. (2017). 12.8 user's manual. *Book 12.8 User's Manual, Series 12.8 User's Manual*.

Duhamel, C., Gouveia, L., Moura, P., e Souza, M. (2008). Models and heuristics for a minimum arborescence problem. *Networks: An International Journal*, 51(1):34–47.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Gurobi, O. (2018). Inc.,"gurobi optimizer reference manual," 2018. *Google Scholar*.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press. ISBN 0472084607.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., e Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical report, Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

https://proceedings.science/p/106895?lang=pt-br

Massen, F., Deville, Y., e Van Hentenryck, P. (2012). Pheromone-based heuristic column generation for vehicle routing problems with black box feasibility. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, p. 260–274. Springer.

Massen, F., López-Ibánez, M., Stützle, T., e Deville, Y. (2013). Experimental analysis of pheromone-based heuristic column generation using irace. In *International Workshop on Hybrid Metaheuristics*, p. 92–106. Springer.

Nepomuceno, N., Pinheiro, P., e Coelho, A. L. (2007a). Tackling the container loading problem: a hybrid approach based on integer linear programming and genetic algorithms. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, p. 154–165. Springer.

Nepomuceno, N., Pinheiro, P., e Coelho, A. L. (2008). A hybrid optimization framework for cutting and packing problems. In *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, p. 87–99. Springer.

Nepomuceno, N. V. (2006). Combinação de metaheurísticas e programação linear inteira: Uma metodologia híbrida aplicada ao problema de carregamento de contêiner. Mestrado, Universidade de Fortaleza, Fortaleza.

Nepomuceno, N. V., Pinheiro, P. R., e Coelho, A. L. (2007b). Combining metaheuristics and integer linear programming: a hybrid methodology applied to the container loading problem. In *Proceedings of the XX congreso da sociedade brasileira de computação, concurso de teses e dissertações*, p. 2028–32.

Philpott, A. B. e Wormald, N. C. (2000). *On the Optimal Extraction of Ore from an Open Cast Mine*. Department of Engineering Science, University of Auckland.

Puchinger, J. e Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, p. 41–53. Springer.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In *International Workshop on Hybrid Metaheuristics*, p. 1–12. Springer.

Rao, V. V. e Sridharan, R. (1992). The minimum weight rooted arborescence problem: Weights on arcs case.

Saraiva, R. D., Nepomuceno, N. V., e Pinheiro, P. R. (2013). The generate-and-solve framework revisited: generating by simulated annealing. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, p. 262–273. Springer.

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Türetken, E., González, G., Blum, C., e Fua, P. (2011). Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, 9(2-3):279–302.

https://proceedings.science/p/106895?lang=pt-br