Uma heurística baseada em GRASP com Path-Relinking para o problema da coloração em grafos esparsos

Igor Lacerda F. da Silva[1], **Vinicius S. Gomes**[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

{igorlfs, vinisilvag}@ufmg.br

Resumo. *Este trabalho propôs o desenvolvimento e comparação de uma heurística híbrida baseada em Greedy Randomized Adaptive Search Procedure (GRASP) com Path-Relinking (PR) para o problema da coloração em grafos esparsos. Como ambas as meta-heurísticas mostraram bons resultados quando aplicadas a outros problemas, o objetivo principal do trabalho foi comparar se o desempenho dessa abordagem híbrida seria competitivo se comparado com soluções modernas (em particular, um algoritmo genético; além do próprio GRASP sem o PR). A partir dos experimentos realizados, foi possível concluir que a heurística híbrida aumentou consideravelmente o tempo de execução, mas não foi capaz de produzir resultados melhores. O link do vídeo de apresentação do trabalho é: https://youtu.be/hXKugzcHZXk.*

1. Introdução

O problema da Coloração de Grafos (*Graph Coloring Problem* ou GCP) pode ser descrito como: dado um grafo simples e não-direcionado $G = (V, E)$ com um conjunto de vértices $V = \{1, 2, \ldots, n\}$ e um conjunto de arestas $E \subset V \times V$, uma k-coloração válida é um mapeamento $c : V \to \{1, 2, \ldots, k\}$ tal que $c(i) \neq c(j)$ para todas as arestas $(i, j) \in E$ [Sun 2018]. Nesse sentido, a k-coloração de grafos (k-GCP) é o problema de determinar se existe uma coloração válida em G que utilize k cores.

Se trata de um problema bastante conhecido e possui muitas aplicações em problemas do mundo real, como na alocação de registradores, na otimização de compiladores [Chaitin et al. 1981], no agendamento de horários de médicos e exames [Lotfi and Sarin 1986], entre outras. No entanto, muitas das aplicações estão interessadas, sobretudo, na versão de otimização do GCP: dado um grafo G, determinar o menor k tal que o grafo G seja k-colorível e qual a coloração atribuída a cada vértice. Esse k também é conhecido como o número cromático $\chi(G)$ do grafo G: o menor número de cores necessárias para colorir G.

Como demonstrado em [Garey and Johnson 1979], o problema da Coloração de Grafos é um problema NP-completo e sua versão de otimização é NP-difícil. Isso significa que não há uma solução exata que execute em tempo polinomial. Como resultado dessa intratabilidade, o uso de heurísticas e meta-heurísticas se torna imprescindível para que o problema e suas instanciações no mundo real possam ser resolvidas.

Além da definição clássica, existem variações do problema. Dentre essas variantes, é possível destacar a Coloração Equitativa de Grafos, a Coloração de Grafos com Vértices Ponderados e a Coloração de Grafos por Largura de Banda, por exemplo. Sendo

assim, avanços nas abordagens de solução de qualquer uma desses variantes podem ser inspiradores para a resolução mais eficiente do problema original (e vice-versa).

De fato, mesmo o GCP clássico pode ser subdivido: é possível restringir o estudo para a solução de uma dada classe de grafos, como grafos com muitas cliques ou grafos cordais. Cada classe possui propriedades específicas que podem ser exploradas na construção de um algoritmo. Além disso, algumas aplicações podem ser modeladas a partir de uma dada classe, de modo que mesmo uma solução para um escopo menor pode ser de utilidade prática. É sob essa ótica que [Laguna and Martí 2001] propõem um algoritmo para o GCP em grafos esparsos (que modelam, por exemplo, o *design* e operação de sistemas de manufatura flexível).

Nesse contexto, esse trabalho, inspirando-se em [Laguna and Martí 2001], propõe o desenvolvimento de uma heurística baseada na meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) em conjunto com a estratégia de intensificação de busca *Path-Relinking* (PR). Ambas as abordagens melhoram conhecidamente os resultados obtidos com relação à qualidade e a robustez da solução e ao tempo de execução [Resende and Ribeiro 2003]. Sendo assim, o objetivo principal é verificar a hipótese de que a aplicação em conjunto dessas abordagens é capaz de levar a soluções melhores (em particular, para grafos esparsos) em tempo semelhante, se comparada a outras abordagens da literatura. Para tanto, a heurística proposta será comparada com outras duas heurísticas, que servirão de *baseline* para determinar o seu desempenho: o GRASP proposto em [Laguna and Martí 2001] e uma heurística baseada em algoritmos genéticos proposta em [Fleurent and Ferland 1996].

A seção 2 discute o estado da literatura em relação à coloração de grafos e outros problemas semelhantes; na seção 3 a metodologia de trabalho é apresentada; na seção 4 os resultados obtidos na experimentação; e na seção 5 o artigo é encerrado.

2. Trabalhos relacionados

Numerosas abordagens foram propostas para resolver o problema da Coloração de Grafos e problemas derivados de maneira aproximada. Dentre elas, as mais conhecidas são as heurísticas construtivas, soluções baseadas em busca tabu e as soluções baseadas em algoritmos evolutivos.

Proposta por [Brélaz 1979], a heurística gulosa DSATUR dá preferência à coloração de vértices com a maior saturação, ou seja, que possuem a maior quantidade de vizinhos já coloridos. Essa heurística se mostrou bastante eficiente e foi muito importante para os avanços seguintes no desenvolvimento de heurísticas para o GCP. Além disso, por ter uma baixa complexidade computacional, ela é utilizada com alguma frequência como ponto de partida para outras meta-heurísticas, como as de busca local, por exemplo.

Após os bons resultados apresentados pelo emprego do *Simulated Annealing* na resolução do GCP, uma solução baseada em busca tabu foi proposta por [Hertz and de Werra 1987], chamada TABUCOL. Os resultados foram bastante positivos e mostraram que o uso de meta-heurísticas, sobretudo a combinação de abordagens em heurísticas mistas, podem levar a resultados ainda melhores que aqueles obtidos com apenas heurísticas construtivas. Essas soluções baseadas em *Simulated Annealing* e busca tabu compõem as meta-heurísticas mais clássicas para o problema.

O estudo de [Fleurent and Ferland 1996] trouxe uma análise completa de alguns algoritmos genéticos e de combinações desses algoritmos com outras meta-heurísticas, especialmente a busca tabu, aplicados ao Problema da Coloração de Grafos. Os resultados obtidos em termos de qualidade da solução foram positivos, no entanto, por serem heurísticas mais lentas, o tempo de execução para as abordagens combinadas se tornou proibitivo em instâncias maiores. Apesar disso, esse trabalho mostrou que a combinação de meta-heurísticas é um caminho promissor, sendo necessário apenas ter cuidado com as heurísticas combinadas, para que o tempo de execução individual não seja longo e isso torne a combinação ineficiente.

O trabalho desenvolvido por [Laguna and Martí 2001] discute o desempenho de algumas implementações de GRASP para o problema da Coloração de Grafos, com ênfase na análise de instâncias com grafos esparsos. Esse trabalho explorou a flexibilidade do GRASP para combiná-lo, também, com uma busca tabu e *Simulated Annealing* na fase de busca local. Apesar de não ser o foco do trabalho, os resultados obtidos para as heurísticas combinadas mostraram que a ideia parece promissora e o balanço entre mais iterações do GRASP e meta-heurísticas mais complexas na fase de busca local deve ser melhor estudado para ser possível tirar o melhor proveito possível da combinação.

Portanto, os esforços empregados na resolução do GCP estão, em sua maioria, direcionados no desenvolvimento de meta-heurísticas para o problema. Tanto o GRASP quanto o *Path-Relinking* são estratégias promissoras, com bons resultados quando aplicadas a outros problemas de otimização [Resende and Ribeiro 2003]. Desse modo, uma abordagem baseada na combinação de ambas as heurísticas parece ser uma alternativa viável para se obter bons resultados em tempo competitivo, em comparação com as heurísticas mais renomadas para o problema.

3. Metodologia

A seguir, as heurísticas implementadas são discutidas, junto de uma descrição da experimentação: quais instâncias foram utilizadas, como o ajuste de parâmetros foi feito e quais foram as principais métricas avaliadas.

3.1. Heurística baseada em algoritmos genéticos

Algoritmos genéticos (*genetic algorithms* ou GA) formam uma classe de meta-heurísticas evolutivas nas quais populações de soluções individuais se desenvolvem inspiradas pela evolução e seleção natural [Fleurent and Ferland 1996]. Em outras palavras, o processo de melhoria do conjunto de soluções corrente é desenhado seguindo a ideia de mecanismos como recombinação gênica, seleção natural e mutações.

Quanto ao algoritmo, inicialmente, um conjunto de N indivíduos é gerado de forma aleatória. Indivíduo, nesse caso, é uma solução válida para o problema. A cada geração, dois indivíduos são selecionados para serem pais e gerarem uma prole. Esse processo é repetido até que NS filhos sejam gerados. Os filhos também possuem uma chance p_m de sofrerem alguma mutação antes de serem inseridos na população. Por fim, o tamanho da população retorna para N através da remoção dos NS piores indivíduos. Quando o número de gerações ($NGen$) é alcançado, o algoritmo termina e o melhor indivíduo observado é retornado.

A codificação das soluções na população foi feita por meio de uma abordagem baseada em *strings*. Isso levou a utilização de estratégias de seleção, *crossover* e mutação inspiradas nas definidas em [Fleurent and Ferland 1996]. A seleção foi feita sorteando dois indivíduos de uma fração da população que possuam um número de cores baixo (indivíduos "mais aptos"). Para o *crossover*, a estratégia utilizada foi a *1-point crossover*, onde uma posição de corte é selecionada aleatoriamente e a prole recebe as cores de um dos pais no lado esquerdo dessa posição e a cor do outro pai no lado direito. Por fim, a mutação será feita definindo que cada nó tem uma probabilidade fixa de ter sua cor trocada por uma outra cor aleatória.

Por ser uma das abordagens mais utilizadas para resolver aplicações do GCP, ela foi escolhida como uma das heurísticas de controle para avaliar o desempenho da heurística proposta. O algoritmo 1 apresenta um pseudocódigo de um GA proposto para o GCP.

Algoritmo 1 Heurística baseada em algoritmo genético

Entrada: $G, N, NGen, NS, p_m$
Saída: $\mathcal{K}, \mathcal{V}^*$

```
P ← ∅
for i ← 1 to N do
    s ← GENERATE-INDIVIDUAL()
    EVALUATE(s)
    APPEND(s, P)
end for
for i ← 1 to NGen do
    for j ← 1 to NS do
        p1, p2 ← SELECT(P)
        offspring ← CROSSOVER(p1, p2)
        MUTATE(offspring, pm)
        EVALUATE(offspring)
        APPEND(offspring, P)
    end for
    REPLACE(P, NS)
end for
```

3.2. Heurística baseada em GRASP

O GRASP é uma meta-heurística que funciona da seguinte maneira: um algoritmo guloso randomizado é usado para gerar uma solução inicial. Em seguida, essa solução inicial é alvo de uma busca local. É importante que esse algoritmo tenha algum componente aleatório, pois a ideia é repetir esse procedimento diversas ($GIter$) vezes para que vários ótimos locais possam ser amostrados (se o algoritmo não fosse randomizado, a mesma solução inicial sempre seria gerada).

No contexto da coloração de grafos, como proposto em [Laguna and Martí 2001], a ideia é construir a coloração uma cor de cada vez, enquanto houver vértices disponíveis. São geradas listas de $CSize$ vértices candidatos (a etapa aleatória é a seleção do candidato). As listas são construídas, inicialmente, olhando-se para os vértices de maior

grau no subgrafo induzido pelo conjunto de vértices que ainda podem receber cor. Uma vez que alguns vértices já não podem mais receber uma atribuição de cor, a geração dos candidatos passa a olhar para os vértices de maior grau (entre os vértices disponíveis) somente considerando os vértices inadmissíveis que ainda não foram coloridos.

Com o candidato escolhido, é criada uma nova cor, e tanto os vizinhos como o próprio candidato são marcados como não disponíveis para atribuição, e os vértices admissíveis também são atualizados. Se a quantidade de arestas restantes for a menor encontrada até então, a nova classe é de fato atribuída para a coloração, uma vez que o processo é repetido $CIter$ vezes. Similarmente, as classes de cores são criadas até todo o grafo estar colorido. Uma vez que isso está concluído, é realizada a busca local.

A busca local inicialmente, tal qual no artigo original, combina as duas menores classes encontradas e tenta remover possíveis inviabilidades fazendo todas as trocas de cores possíveis. A solução corrente é atualizada se o número de arestas proibidas diminui. O pseudocódigo dessa etapa é omitido na apresentação a seguir, do algoritmo 2.

3.3. Heurística proposta (GRASP + PR)

A heurística proposta pelo trabalho também se baseia num GRASP, no entanto, ela o combina com uma estratégia de intensificação de busca conhecida como *Path-Relinking*. Descrita originalmente por [Glover 1996], o *Path-Relinking* é uma abordagem que explora soluções intermediárias entre duas soluções, chamadas de solução inicial e solução guia. A ideia é que, explorando as soluções intermediárias, é possível encontrar soluções melhores que as soluções obtidas até o momento.

Para realizar esse processo, a diferença simétrica (Δ) entre a solução inicial e a solução guia é calculada, ou seja, quais mudanças devem ocorrer na solução inicial para que ela se torne igual à solução guia (quais vértices devem mudar de cor). Após isso, cada mudança (Δ_j) é aplicada à solução inicial e o novo custo é calculado (o novo número de cores utilizadas). Em caso de melhora, a solução atual é atualizada para essa nova solução aprimorante. Esse processo é repetido até que todas as mudanças sejam aplicadas.

Caso uma mudança Δ_j transforme a solução inicial numa solução inválida, o teste que verifica se a solução intermediária é melhor que a atual será pulado até que uma nova mudança transforme a solução intermediária em uma solução válida novamente.

A escolha das soluções inicial e guia acontece do seguinte modo: o PR analisa as n melhores soluções retornadas pelo GRASP, chamadas de soluções de elite (X). A melhor solução é escolhida como a solução guia e as outras $n - 1$ são comparadas com essa solução. Ao final, a melhor solução retornada será a melhor entre o PR e pela heurística. O algoritmo 3 apresenta um pseudocódigo para a heurística proposta.

3.4. Planejamento dos experimentos

Os experimentos foram inspirados nos procedimentos realizados em [Laguna and Martí 2001], uma vez que este foi o trabalho que originalmente propôs o GRASP para este problema. Foram tratadas duas classes de problemas: grafos gerados artificialmente[1], com solução (número cromático) conhecida, e grafos randomizados.

[1]É possível realizar o *download* dessas instâncias neste link.

Algoritmo 2 Heurística baseada em GRASP

Entrada: $V, E, GIter, CIter, CSize, \mathcal{K}, \mathcal{V}^*$
Saída: $\mathcal{K}, \mathcal{V}^*$

```
K ← |V|
for iter ← 1 to GIter do
    i ← 0
    V' ← V
    while V' ≠ ∅ do
        i ← i + 1
        ecount ← ∞
        for j ← 1 to CIter do                ▷ Início da construção gulosa randomizada
            V̂ ← V'
            U ← ∅
            C ← ∅
            while V̂ ≠ ∅ do
                if U = ∅ then
                    CL ← {CSize vértices de máximo grau em V̂}
                else
                    CL ← {CSize vértices em V̂ de máximo grau em U}
                end if
                Selecione algum v ∈ CL aleatoriamente
                C ← C ∪ {v}
                N(v) ← {w | (v, w) ∈ E}
                V̂ ← V̂ − {v} − N(v)
                U ← U ∪ N(v)
            end while
            E* ← {(u, v) ∈ E | u, v ∈ V' − C}
            if |E*| < ecount then
                V_i ← C
                ecount ← |E*|
            end if
        end for
        V' ← V' ∪ V_i
    end while
    LOCAL-SEARCH(V, K)                       ▷ Fase de busca local
    if i < K then                            ▷ Atualiza a melhor solução
        V ← {V_1, ..., V_i}
        K ← i
    end if
end for
```

Os grafos artificiais, por sua vez, se dividem em 4 subclasses: LEI (Grafos Leighton, ver [Leighton 1979]), MYC, REG e SGB. O propósito dessas instâncias artificiais é fornecer um parâmetro razoável do desempenho dos algoritmos, uma vez que o número cromático é conhecido. As subclasses estão divididas da seguinte maneira:

- 12 instâncias LEI, com número de vértices igual a 450;

Algoritmo 3 Heurística combinada GRASP + PR

Entrada: $V, E, GIter, CIter, CSize, \mathcal{K}, \mathcal{V}^*$
Saída: $\mathcal{K}, \mathcal{V}^*$

```
X ← GRASP-COLOR(V, E, GIter, CIter, CSize, K, V*)   ▷ X são as soluções elite
s_g ← BEST-SOLUTION(X)                               ▷ Seleciona a solução guia
while X ≠ ∅ do
    s_i ← POP(X)                                     ▷ Seleciona uma solução inicial
    Δ ← SYMMETRIC-DIFFERENCE(s_i, s_g)               ▷ Calcula a diferença simétrica
    while Δ ≠ ∅ do
        s_inter ← CHANGE-COLORING(s_i, Δ_j)          ▷ Troca a cor do vértice j ∈ Δ
        (V_inter, K_inter) ← s_inter
        if K_inter < K then                          ▷ Atualiza a melhor solução
            V* ← V_inter
            K ← K_inter
        end if
        Δ ← Δ \ {Δ_j}                                ▷ Remove o movimento Δ_j de Δ
    end while
end while
```

- 5 instâncias MYC;
- 14 instâncias REG;
- 10 instâncias SGB.

Essas instâncias foram escolhidas, principalmente, por serem compostas por grafos esparsos e por serem frequentemente adotadas na literatura para o problema (inclusive em [Laguna and Martí 2001]).

A segunda bateria de testes, é composta por 400 grafos aleatórios ($G_{n,p}$). São 100 instâncias para as quantidades de vértices (n) iguais a 50, 100, 250 e 500. A geração desses grafos foi feita a partir de variáveis aleatórias x_{ij} (distribuídas uniformemente em [0,1]) para cada par (não ordenado) $\{i, j\} \in \{1, \ldots, n\}$. Quando x_{ij} é menor que a probabilidade p, então existe aresta de i para j. Como o grafo é simples, as probabilidades x_{ii} são nulas. Como os grafos em questão são esparsos, o p escolhido foi $\frac{1}{10}$.

A primeira fase da experimentação foi realizar o ajuste dos hiper-parâmetros das heurísticas, com base nas instâncias artificiais. Para o algoritmo genético, alguns parâmetros foram fixados, graças aos bons resultados observados em [Fleurent and Ferland 1996], enquanto outros foram variados:

$$N = 50.000, 80.000, 120.000, 150.000$$
$$NGen = 50, 100, 150$$
$$NS = 2$$
$$p_m = 0.01$$

Enquanto que para o GRASP:

$$GIter = 5, 15, 25$$
$$CIter = 5, 15, 25$$
$$CSize = 3, 6, 9$$

Por fim, com o *Path-Relinking* também foram analisados:

$$|X| = 5, 10, 15$$

Para avaliar o desempenho das heurísticas, os principais critérios analisados foram o número de cores usadas na melhor solução e o tempo despendido na execução de cada algoritmo. Além disso, todos os testes, tanto para o ajuste de parâmetros quanto para a realização dos experimentos, foram executados em um computador com processador Intel i5-10210U (8) @ 4.200GHz e 8 GB de RAM.

4. Resultados

Na etapa inicial da experimentação, devido às restrições de tempo, a escolha dos parâmetros foi realizada com base em uma única execução de cada combinação de parâmetros. Como a variância entre essas execuções foi baixa, avalia-se que essa decisão não há de ter causado um grande impacto na escolha dos parâmetros.

Para o algoritmo genético, os parâmetros obtidos foram $N = 80.000$ e $NGen = 100$. Para o GRASP: $GIter = 25, CIter = 25, CSize = 3$, enquanto que para o Path-Relinking, foram escolhidas as 10 melhores soluções do GRASP para análise. Os valores obtidos nesta fase foram utilizados no restante da análise experimental.

Ao executar as heurísticas contra as instâncias artificiais, vemos nas tabelas 1 e 2, as quantidades de cores médias que foram usadas em cada abordagem e o tempo médio despendido executando todas as instâncias, respectivamente. Como essas instâncias possuem número cromático conhecido, é possível avaliar o desempenho das heurísticas em relação ao ótimo.

Subclasse	Instâncias	$\bar{n}$	$\bar{m}$	$\bar{\chi}$	GA	GRASP	GRASP + PR
LEI	12	450.0	110.005	15.0	63.83	17.0	17.16
MYC	5	73.4	688.4	6.0	6.6	6.0	6.0
REG	14	362.1	7608.1	37.4	81.64	37.41	37.41
SGB	10	113.9	2835.2	22.6	23.5	22.7	22.7
Total / Média	41	249.9	5534.3	20.2	43.89	20.76	20.80

Tabela 1. Quantidade média do número de cores utilizadas

Subclasse	GA	GRASP	GRASP + PR
LEI	44459	98135	605822
MYC	3983	2703	3651
REG	29861	27890	192803
SGB	5427	1560	7074
Total / Média	20933	32572	202337

Tabela 2. Tempo médio de execução, em milissegundos

Analisando o número médio de cores, é possível perceber que o desempenho do algoritmo genético não foi tão bom quanto esperado, especialmente para as subclasses LEI e REG, que possuem as instâncias com os maiores números de vértices. Por outro lado, o desempenho do GRASP foi tão bom que não pôde ser superado com o uso do

Path-Relinking, apresentando apenas uma pequena variação na média devido ao caráter randomizado dos algoritmos.

Além disso, analisando os tempos médios observados, foi verificado que o uso *Path-Relinking*, além de não ter melhorado a qualidade das soluções, introduziu um custo significativo no tempo de execução. O GA, por sua vez, apresentou um tempo relativamente baixo. No entanto, por se tratar de uma classe de heurísticas nas quais é possível aumentar o número de iterações arbitrariamente, na busca por soluções mais aprimorantes, um dos critérios utilizados para a escolha dos parâmetros foi que o tempo computacional despendido não fosse proibitivo, visando manter a análise mais balanceada.

Assim, um dos resultados esperados não foi obtido: não foi possível aprimorar o GRASP com o uso do *Path-Relinking* para as instâncias artificiais. De fato, o GRASP + PR retornou soluções muito próximas das soluções obtidas para o GRASP, mas o PR acrescentou um custo proibitivo no tempo de execução do algoritmo, em especial para as instâncias com um número de vértices maior.

Em seguida, os mesmos experimentos foram realizados para as instâncias aleatórias. Os resultados foram agregados nas tabelas 3 e 4.

n	$\bar{m}$	GA	GRASP	GRASP + PR
50	122.91	4.92	3.95	3.99
100	491.84	7.23	5.22	5.55
250	3114.04	17.37	9.35	9.55
500	12490.64	57.05	15.02	15.14
Média	4054.86	21.64	8.39	8.55

Tabela 3. Quantidade média do número de cores utilizadas nos grafos $G_{n,0.1}$

n	GA	GRASP	GRASP + PR
50	3377	26	58
100	9006	1531	1048
250	22343	14654	53564
500	76906	230845	791902
Média	27908	61764	211643

Tabela 4. Tempo médio de execução, em segundos

Da mesma forma como foi observado para as instâncias artificiais, não há uma diferença significativa entre a qualidade dos resultados entre o uso somente do GRASP em comparação com o GRASP + PR. Novamente o GA não obteve o desempenho desejado, possuindo resultados competitivos apenas para as instâncias menores, mas empregando um número de cores bem maior para as instâncias com muitos vértices.

Em termos de tempo de execução, a melhoria proposta causou um aumento proibitivo, se comparado com as heurísticas de controle. Além disso, em comparação com o GA, o tempo de execução perde a completividade conforme o tamanho das instâncias aumenta.

5. Considerações finais

De modo geral, foi constatado que a modificação proposta para a heurística introduzida por [Laguna and Martí 2001] não trouxe bons resultados para grafos esparsos: o número de cores empregado não diminuiu e o custo computacional aumentou consideravelmente, podendo ser proibitivo em alguns casos.

Entretanto, isso pode estar relacionado com o fato dos grafos serem esparsos. Uma pesquisa futura poderia verificar o desempenho do *Path-Relinking* para grafos mais densos e avaliar se esta característica favorece essa estratégia, permitindo que soluções mais variadas possam ser observadas e comparadas.

Por fim, o desempenho do PR pode estar relacionado à possibilidade da solução guia e as soluções iniciais serem iguais, a menos de uma permutação nas cores. Sendo assim, um trabalho futuro poderia explorar a possibilidade de normalizar as soluções antes da aplicação do PR, na tentativa de otimizar sua aplicação e o tempo computacional gasto.

Referências

[Brélaz 1979] Brélaz, D. (1979). New methods to color vertices of a graph. *Commun. ACM*, 22(4):251–256.

[Chaitin et al. 1981] Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E., and Markstein, P. W. (1981). Register allocation via coloring. *Comput. Lang.*, 6(1):47–57.

[Fleurent and Ferland 1996] Fleurent, C. and Ferland, J. A. (1996). Genetic and hybrid algorithms for graph coloring. *Ann. Oper. Res.*, 63(3):437–461.

[Garey and Johnson 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[Glover 1996] Glover, F. (1996). Tabu search and adaptive memory programing: Advances, applications, and challenges. In Barr, R., Helgason, R., and Kennington, J., editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer.

[Hertz and de Werra 1987] Hertz, A. and de Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351.

[Laguna and Martí 2001] Laguna, M. and Martí, R. (2001). A GRASP for coloring sparse graphs. *Comput. Optim. Appl.*, 19(2):165–178.

[Leighton 1979] Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *J. Res. Nat. Bur. Standard*, 84(6):489–506.

[Lotfi and Sarin 1986] Lotfi, V. and Sarin, S. (1986). A graph coloring algorithm for large scale scheduling problems. *Comput. Oper. Res.*, 13(1):27–32.

[Resende and Ribeiro 2003] Resende, M. G. C. and Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In Glover, F. W. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 219–249. Kluwer / Springer.

[Sun 2018] Sun, W. (2018). *Heuristic Algorithms for Graph Coloring Problems. (Algorithmes heuristiques pour des problèmes de coloration de graphes)*. PhD thesis, University of Angers, France.