ESTUDO SOBRE ALGORITMOS PARA RESOLUÇÃO DE CONJUNTO INDEPENDENTE NA DEFINIÇÃO DE EQUIPES DE EMPREENDEDORES

Felipe Simoni Dalcin[1], Rafael de Santiago[2]

[1]Curso de Ciência da Computação – Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360 – CEP 88302-202 – Itajaí – SC – Brasil

{felipe.dalcin, rsantiago}@univali.br

Abstract. *TUTOR is an application that aims to provide incorporation of entrepreneurial characteristics and self-knowledge to users. Thus, is proposed the suggestion of groups where low competence characteristics must be mitigated by other users in the group. This process is related to Independent Set problem and requires exponential time. In this context, a heuristic algorithm was developed based on GLP heuristic and presented good results, finding best solutions in polynomial time frequently. Despite the small amount of data in TUTOR, the algorithm is able to suggest groups with a few undesirable cases.*

Resumo. *A ferramenta TUTOR busca proporcionar a incorporação de características empreendedoras e autoconhecimento aos usuários. Desta forma, é proposta a sugestão grupos de empreendedores onde características de baixa intensidade de cada indivíduo sejam amenizadas por outros empreendedores da equipe. Este processo está relacionado ao problema do Conjunto Independente e demanda tempo exponencial. Neste contexto, foi desenvolvido um algoritmo heurístico, baseado na heurística GLP, que apresentou bons resultados, encontrando as melhores soluções conhecidas em tempo polinomial. Apesar do pouco volume de dados no TUTOR, o algoritmo foi capaz de sugerir grupos, apresentado poucos casos indesejáveis.*

1. Introdução

McClelland (1972) mostrou que empreendedores de sucesso possuem algumas características comportamentais que os diferem dos demais. Estas foram organizadas em três grandes grupos, sendo eles: (*i*) Conjunto de realização; (*ii*) Conjunto de planejamento; e (*iii*) Conjunto de poder.

Neste contexto, a ferramenta TUTOR, proposta por Pessoa (2011), tem por objetivo monitorar e proporcionar a incorporação de tais características nos usuários, possibilitando a compreensão das mesmas e relacionando-as com atividades cotidianas. Desta forma, a ferramenta proporciona autoconhecimento aos usuários, o qual é a base para o processo de desenvolvimento. Sendo assim, o TUTOR fornece ao usuário a compreensão de características fortes e fracas em seu comportamento, permitindo que o mesmo trabalhe para compensar vulnerabilidades.

Este trabalho propõe a ampliação da ferramenta TUTOR, desenvolvendo uma funcionalidade que possibilite a sugestão de equipes de empreendedores com características complementares, de modo a amenizar os pontos fracos de cada

empreendedor. Desta forma, seria possível a sugestão de grupos qualificados e equilibrados quanto as características empreendedoras. No entanto, esta sugestão de grupos é considerada computacionalmente difícil, pois está relacionada à um problema que demanda tempo exponencial, necessitando de muito tempo para possibilitar a verificação dos resultados.

O objetivo é possibilitar a agregação desta funcionalidade, de modo que os resultados sejam obtidos em tempo polinomial. Para cumprir com este objetivo, utilizou-se de soluções heurísticas para o Conjunto Independente ou Clique, classificados na literatura como NP-Completo de difícil aproximação.

Na sequência deste artigo são apresentados tópicos referentes às características complementares, Conjunto Independente/Clique, heurística (sendo dividida em pré-processamento, processamento e pós-processamento), resultados e conclusões.

2. Características complementares

A identificação de características que possam ter consequências negativas para o empreendedor foi importante, de modo a determinar a classificação dos grupos gerados pela funcionalidade. Segundo Rosa e Morales (2010), é possível estabelecer uma relação entre as características, de forma que a ausência de alguma delas, pode tornar um empreendedor vulnerável. A relação entre as características combinadas com suas intensidades permite antever obstáculos.

Quadro 1. Principais combinações que podem tornar o empreendedor vulnerável.

Intensidade		**Consequência**
Alta	**Baixa**	
Metas	Persistência	Desiste da meta perante obstáculos significativos.
Persistência	Metas	Tende a ser teimoso, demorando em buscar alternativas para alcançar os objetivos.
Riscos	Busca de informações	Pode tomar decisões baseado em suposições, deixando de correr riscos calculados e apenas correndo riscos.
Planejamento	Busca de informações	Elaboração de planos a partir de hipóteses e dados insuficientes. Não buscando informações sobre resultados da execução do plano não será alertado sobre mudanças necessárias no mesmo.
Planejamento	Metas	Tendência a acreditar que um conjunto de atividades trará resultados positivos, porém como não tem clareza de quais são esses resultados, o mesmo não tem condições de propor ações corretivas ao monitorar resultados parciais.
Qualidade	Comprometimento	Em algumas situações pode não cumprir padrões de qualidade combinados ao perceber que isso significará fazer sacrifícios (financeiros ou não).
Autoconfiança	Metas	Estabelece algumas metas irreais, baseadas na ideia distorcida de que o empreendedor pode fazer tudo e não na sua capacidade real.
Persuasão	Comprometimento	Pode prometer o que não pode cumprir para convencer clientes e fornecedores, mesmo sabendo que não há possibilidade de honrar tais compromissos.

Rosa e Morales (2010) estabeleceram as principais combinações de características que podem afetar negativamente o desempenho de um empreendedor. Estas combinações podem ser visualizadas no Quadro 1.

A partir da identificação destas combinações de comportamentos em um empreendedor, é possível que o mesmo adquira autoconhecimento, buscando a equalização do perfil em empreendedores de perfil complementar, melhorando assim seu desempenho.

3. Conjunto Independente/Clique

Em um grafo $G = (V, E)$, um Conjunto Independente I se define por um subconjunto de V, tal que todos os vértices de I são pares não adjacentes, ou seja, não há arestas em E ligando tais vértices. De maneira mais formal, I é um Conjunto Independente, sendo $I = \{i \in V \mid (i, v) \notin E, \forall v \in I\}$. Este problema possui variações, sendo: (*i*) Conjunto Independente Máximo (CIM), o qual consiste em encontrar o Conjunto Independente com a maior quantidade de vértices em G; e (*ii*) k-Conjunto Independente, que procura saber se um grafo possui um Conjunto Independente de tamanho específico k (BATSYN et al., 2013).

Segundo Butenko e Wilhelm (2006), o Conjunto Independente é estreitamente ligado a outro problema da classe NP-Completo, conhecido como Clique. O Clique de um grafo G é um subgrafo completo de G, ou seja, os vértices do conjunto possuem todas as ligações possíveis. Da mesma forma que o Conjunto Independente, o Clique possui variações, sendo o Clique Máximo (CM) que consiste em encontrar um clique com maior número de vértices em G, enquanto um k-Clique limita-se a um tamanho específico k, sendo k o número de vértices do Clique [Sipser, 2007] [Bomze 1999].

Segundo Garey e Johnson (1979), Clique e Conjunto Independente são formas diferenciadas de se olhar para um mesmo problema. Os autores mostraram que soluções aplicadas à um problema de Clique, também se aplicam à problemas do Conjunto Independente. Isto é possível porque um Conjunto Independente de um grafo G é representado por um Clique no complemento G^c de G, conforme Figura 1, onde pode ser verificado um grafo G (esquerda) com um CIM (Conjunto Independente Máximo) destacado pelos vértices $\{A, C, E\}$ e um grafo G^c complementar de G (direita) com um CM (Clique Máximo) destacado pelos mesmos vértices $\{A, C, E\}$. Sendo assim:

1. I é um Conjunto Independente para $G = (V, E)$;
2. I é um Clique no grafo complementar G^c de G, onde $G^c = (V, E^c)$, sendo $E^c = \{(u, v): u, v \in V \; e \; (u, v) \notin E\}$, ou seja, G^c possui todas as ligações não presentes em G.

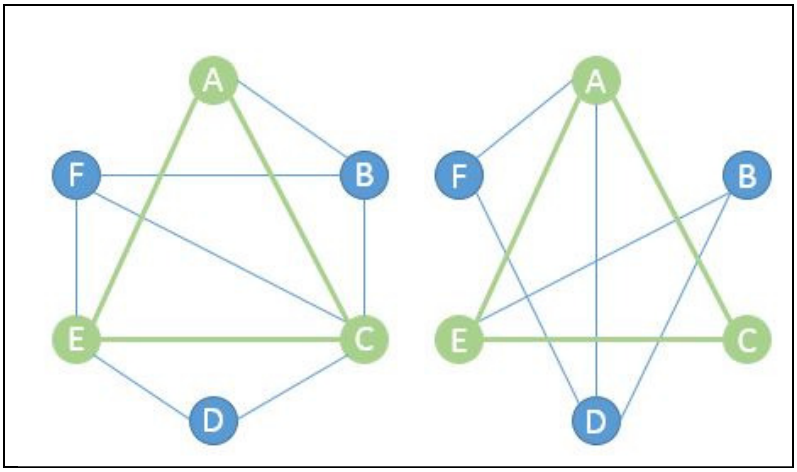


Figura 1. CIM no Grafo G (esquerda) e CM no Grafo G^c (direita).
Fonte: Adaptado de Rese e Santiago (2012).

4. Heurística

Grosso, Locatelli e Pullan (2007) apresentaram uma heurística bastante eficiente para solucionar o problema do Clique. A proposta consiste em um algoritmo de Iterated Local Search, onde as características principais são: (*i*) rápida busca em vizinhança, baseadas em seleções completamente aleatórias e não na avaliação dos vértices; e (*ii*) técnicas de diversificação e regras de reinício simples.

A seleção dos vértices para as operações do algoritmo é feita de forma aleatória, pois segundo Grosso, Locatelli e Pullan (2007), técnicas de avaliação de vértices candidatos são, de certa forma, cegas e facilmente enganadas em instâncias maiores do Clique. Além disto, a heurística se utiliza de regras de reinício, definindo de forma eficaz o espaço de busca do algoritmo.

O algoritmo desenvolvido neste trabalho teve como base a heurística GLP, proposta por Grosso, Locatelli e Pullan (2007). Este sofreu modificações para possibilitar a busca de k-cliques em um grafo G. Também optou-se por fixar o primeiro elemento da solução, garantindo ao menos uma solução para cada vértice V de G.

O processo de sugestão de equipes se dividiu em três etapas distintas, sendo elas: (*i*) pré-processamento; (*ii*) processamento; e (*iii*) pós-processamento. Estas etapas são apresentadas a seguir.

4.1 Pré-processamento

A etapa de pré-processamento teve por objetivo a busca das informações necessárias para a construção do grafo. Este é composto pelo conjunto de empreendedores, ligados uns aos outros através das características de baixa intensidade em comum. O resultado desta etapa consiste em um grafo $G = (V, E)$. No entanto, como o objetivo é encontrar o Conjunto Independente para estas instâncias, foi aplicada uma transformação polinomial de modo a obter o grafo complementar $G^c = (V, E^c)$.

Estas informações foram armazenadas em um arquivo, conforme Figura 2, similar às instâncias da família DIMACS, padronizando a leitura e criação dos grafos processados pela heurística. O arquivo apresenta campos como: (*i*) "c FILE:" indica o nome do arquivo; (*ii*) linhas iniciadas com a letra "v" identificam o vértice e a qual

empreendedor este se refere; (*iii*) "p edge" identifica, respectivamente o número de vértices e de arestas do grafo; e (*iv*) linhas iniciadas em "e" informa as arestas presentes no grafo.

```
c FILE: grafo tutor.clq
c Graph Size:100
v 1 - 247 Empreendedor 1
...
v 100 - 346 Empreendedor 100
p edge 100 2557
e 1 4
e 1 9
e 1 11
...
e 97 99
e 99 100
```

Figura 2. Exemplo de grafo TUTOR.

4.2 Processamento

A etapa de processamento abrange o algoritmo heurístico desenvolvido. Inicialmente foi codificada, na linguagem C++, uma versão da heurística GLP, a qual foi testada e comparada à heurística original a fim de verificar a aceitação do algoritmo. Os testes se utilizaram de instâncias da família DIMACS e apresentaram resultados próximos aos relatados no artigo de Grosso, Locatelli e Pullan (2007), embora tenha consumido tempo superior (Tabela 1). Nesta tabela, para cada instância é apresentado o número de vértices (n), o número de arestas (m), a melhor solução conhecida (ω(G)), a solução média obtida (CLQavg), seguido pelo menor e maior valor encontrado, e o tempo médio em segundos (t(s)).

Tabela 1. Resultados GLP Original x GLP Implementado.

Grafo				GLP Original		GLP Implementado	
Nome	**n**	**m**	**ω(G)**	**CLQavg**	**t(s)**	**CLQavg**	**t(s)**
brock200_1	200	14834	21	21	0,002	21	0,061
brock200_2	200	9876	12	12	0,04	12	0,43
brock200_3	200	12048	15	15	0,25	15	5,21
brock200_4	200	13089	17	17	0,13	17	2,99
brock400_1	400	59723	27	26,28 (25, 27)	130,77	26,5 (25, 27)	4772
brock400_2	400	59786	29	29	29,53	29	1252
brock400_3	400	59681	31	31	3,83	31	129,31
brock400_4	400	59765	33	33	1,83	33	51,67

No entanto, para permitir a sugestão de grupos, foram adicionadas algumas modificações de modo a encontrar vários grupos de tamanho k, ou seja, k-cliques ao invés do Clique Máximo do grafo. Além disto, a cada execução, o primeiro elemento da

solução foi fixado, permitindo encontrar um grupo para cada vértice (empreendedor) de um grafo $G = (V, E)$, de modo que ao final existam $|V|$ grupos.

Os resultados obtidos pelo algoritmo foram armazenados em um arquivo, exemplificado na Figura 3, sendo: (*i*) linhas iniciadas com a letra "v" identificam vértices e sua relação com o código do respectivo empreendedor; e (*ii*) as linhas iniciadas com a letra "g" identificam os grupos gerados, sendo os vértices separados por vírgula.

```
v 1 - 247
...
v 100 - 346
g 1,78,64,27,25,35,49,37,67,62
g 11,90,64,30,63,94,13,27,9,88
...
g 92,95,84,39,4,9,3,73,81,64
g 100,17,50,40,53,67,27,8,12,21
```

Figura 3. Exemplo de Grupos Resultantes.

4.3 Pós-processamento

A etapa de pós-processamento reuniu as informações da etapa de processamento, de modo a analisar e encontrar possíveis vulnerabilidades identificadas por Rosa e Morales (2010). Grupos que apresentaram tais problemas foram descartados, garantindo grupos mais qualificados quanto a combinação das características empreendedoras.

Os grupos são analisados de modo a identificar se para todas as características há um integrante com alta competência, sendo estes considerados ótimos. Caso haja alguma característica que não possua integrante com alta intensidade, o grupo é analisado buscando identificar as vulnerabilidades destacadas por Rosa e Morales (2010). Os grupos são considerados indesejáveis caso seja encontrada alguma destas fraquezas. Caso não seja identificada nenhuma destas vulnerabilidades o grupo é considerado desejável. Desta forma, pode-se destacar três possíveis resultados, sendo eles:

- Grupos ótimos: esta é a solução ótima e esperada, pois são grupos em que todas as características são compensadas por pelo menos um integrante;
- Grupos desejáveis: são soluções próximas da ótima. Eventualmente, alguma característica não é compensada por membros do grupo. No entanto, a combinação das características não é considerada uma real vulnerabilidade;
- Grupos indesejáveis: são soluções que não atingem bons resultados. Estes grupos são similares aos desejáveis. No entanto, a combinação das características pode gerar uma ou mais vulnerabilidades identificadas por Rosa e Morales (2010).

5. Resultados

Foram realizados testes a fim de verificar a aceitação do algoritmo para o Clique Máximo e seus resultados foram comparados à heurística original, conforme

apresentado na Tabela 1. A partir dos resultados apresentados, é possível perceber que o tempo de execução da heurística original é mais eficiente em relação à desenvolvida neste trabalho. Este contraste se dá, possivelmente, pelas estruturas e detalhes de implementação que se diferem. No entanto, o algoritmo encontrou as melhores soluções conhecidas, pelo menos uma vez, para todas as instâncias testadas. O gráfico da Figura 4, permite uma visualização mais clara da diferença de tempo entre os algoritmos.

Os resultados apresentados por Grosso, Locatelli e Pullan (2007) mostram que a instância brock400_1 apresentou complexidade de tempo consideravelmente superior em relação às outras instâncias. Isto também foi observado nos experimentos realizados para este trabalho. Apesar da diferença de tempo em relação à heurística original, a implementação do algoritmo GLP foi capaz de encontrar as melhores soluções conhecidas em tempo polinomial, possibilitando sua utilização para cumprir com o objetivo principal deste trabalho.

Também foram realizados testes experimentais em 75 instâncias de uma família mista, possuindo variações quanto as densidades, entre 10% e 100%, e número de vértices $n = \{128,256,1024,2048,4096,8192,16384\}$. Estes testes buscaram encontrar 10-cliques, ou seja, cliques de tamanho 10.

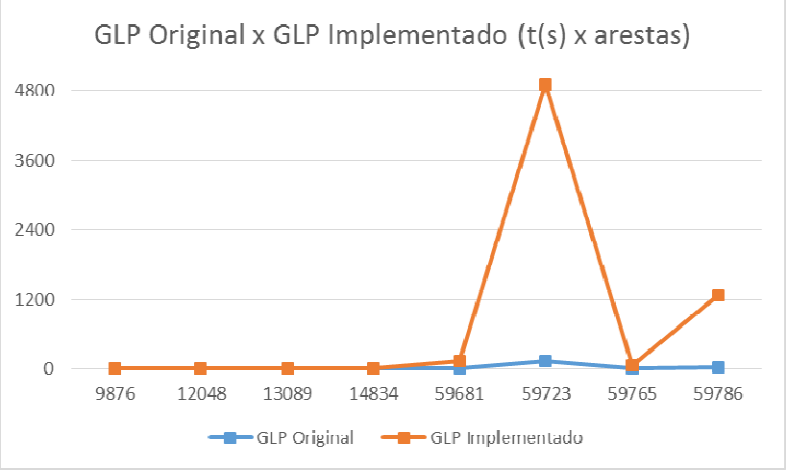
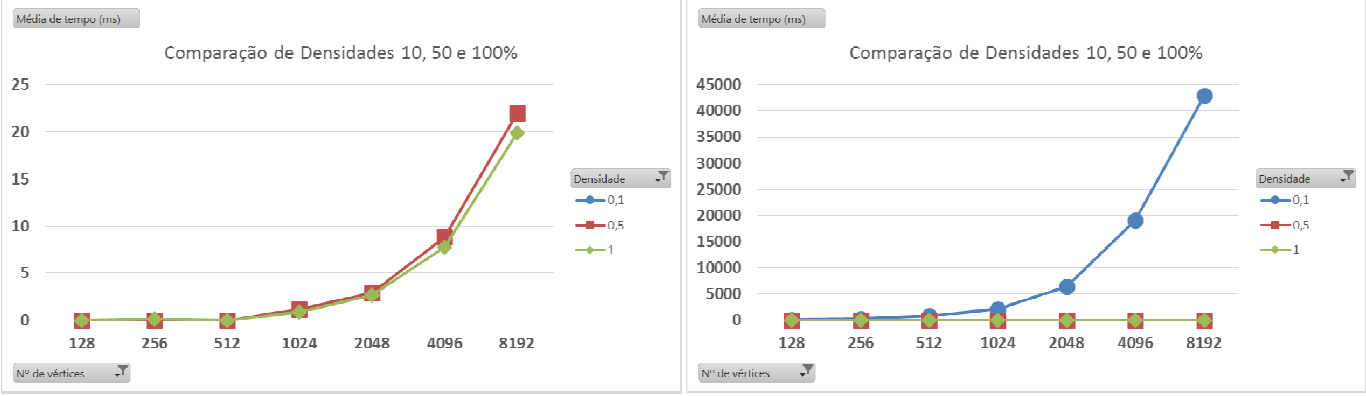


Figura 4. Comparação GLP Original x GLP Implementado.



(a) **(b)**

Figura 5. Comparação de Tempo de Execução Para Diferentes Densidades.

A Figura 5 representa graficamente os resultados obtidos, onde é possível identificar que o algoritmo demanda maior complexidade computacional para instâncias esparsas, em que não há muitas conexões entre os vértices. Para estas instâncias o algoritmo dificilmente encontra um 10-clique, pois pode não haver tal clique no grafo.

Testes foram feitos para validar a aderência a ferramenta TUTOR, estes se utilizaram de dados fictícios. Foram gerados 100 usuários e para cada um deles foi formado um conjunto de características, aleatoriamente. Estas foram distribuídas respeitando um percentual para cada intensidade, sendo: (*i*) 25% para alta intensidade; (*ii*) 35% para média intensidade; e (*iii*) 40% para baixa intensidade.

A Tabela 2 apresenta um grupo obtido durante os testes realizados, onde a coluna de preenchimento cinza indica que não há empreendedor com alta intensidade na respectiva característica. Representando um grupo onde não há alta intensidade para todas as competências, no entanto a combinação das mesmas não gera uma vulnerabilidade.

5.1 Regressão Linear

A regressão linear foi realizada para estimar a função de complexidade amortizada para o algoritmo. Para tal, foi utilizada a ferramenta GNU R, que consiste em um ambiente de software para cálculos de computação estatística e gráficos. Desta forma, foi possível obter a relação entre quantidade de vértices, arestas e o tempo computacional resultante dos testes.

A Tabela 3 apresenta os resultados obtidos através da regressão linear para cada modelo, onde: (*i*) "Modelo" se refere ao modelo de regressão submetido; (*ii*) "Resultado" indica a função de complexidade resultante; e (*iii*) "Erro Residual" consiste na distância entre os dados experimentais em relação ao que é previsto pela função obtida.

Considera-se como complexidade experimental do algoritmo a função $\Theta(n^{(6,918)}\, m^{(-3,044)})$, descartando-se e, pois representa a constante equivalente a infraestrutura/tecnologia utilizada durante os experimentos. A escolha desta função se deve ao menor erro residual encontrado entre os modelos utilizados. A análise da função corrobora com os resultados exibidos anteriormente: quanto maior o número de arestas, menor a demanda de tempo computacional.

6. Conclusão

A ferramenta TUTOR, proposta por Pessoa (2010), tem por objetivo monitorar e proporcionar a incorporação de características do comportamento empreendedor nos usuários que a utilizem. No entanto, segundo Rosa e Morales (2010), um empreendedor, se agindo sozinho, pode se tornar vulnerável em relação às características comportamentais. Desta forma, o objetivo principal deste trabalho foi desenvolver uma funcionalidade capaz de sugerir equipes de empreendedores com características complementares, compensando eventuais vulnerabilidades individuais. Para cumprir com este objetivo, este trabalho analisou problemas de Conjunto Independente e Clique, classificados na literatura como NP-Completo de difícil aproximação.

A solução proposta neste trabalho se baseou na heurística GLP, apresentada por Grosso, Locatelli e Pullan (2007), a qual trata de soluções para o Clique Máximo. A

versão da GLP implementada foi analisada, de modo a verificar sua aceitação. Posteriormente, esta sofreu modificações para possibilitar a análise de soluções de k-cliques, de modo a encontrar vários grupos de tamanho k em um grafo G.

A versão da heurística GLP foi testada com instâncias da família DIMACS e apresentou tempos superiores aos relatados por Grosso, Locatelli e Pullan (2007). Este contraste ocorre, possivelmente, por diferenças nas estruturas e codificação. No entanto, a versão implementada foi capaz de atingir as melhores soluções conhecidas em tempo polinomial, o que garantiu sua aceitação para o projeto.

A implementação GLP para k-cliques, foi testada com uma família mista de 75 grafos e apresentou ótimos resultados em relação à sua complexidade, garantindo cliques de tamanho k em tempo bastante satisfatório. A partir dos testes, foi possível perceber que o algoritmo apresentou maior tempo computacional para grafos esparsos, onde há menor número de arestas entre os vértices, dificultando a busca por k-cliques.

Testes relacionados à qualidade apresentaram bons resultados, garantindo, em sua maioria, grupos em que todas as características são compensadas por pelo menos um empreendedor. No entanto, os testes foram realizados com dados fictícios, buscando simular um grupo real de pessoas. Desta forma, os resultados não possuem volume suficiente de dados para considerar a solução como definitiva, havendo possibilidades de melhoria na complexidade de tempo, bem como na qualidade da solução.

Tabela 2. Grupo Resultante.

Nome	Busca de Inf.	Busca de Op.	Comp.	Correr Riscos	Qual.	Persis.	Metas	Plan.	Persu.	Ind.
Abigail Tabares	B	B	M	M	***A***	B	M	***A***	B	B
Diana Gómez	M	***A***	B	***A***	B	***A***	M	***A***	B	***A***
Edmundo Carballo	***A***	***A***	***A***	B	***A***	***A***	M	B	***A***	B
Flávia Carrasqueira	B	M	M	B	B	M	B	***A***	***A***	B
Gertrudes Passos	***A***	***A***	B	B	M	B	B	***A***	B	M
Israel Pari	B	***A***	B	M	B	B	M	***A***	***A***	M
Mateus Cotrim	M	M	B	M	***A***	M	M	***A***	M	B
Melinda Picanço	B	M	***A***	M	***A***	***A***	M	***A***	***A***	***A***
Nazaré Gomes	B	B	B	***A***	M	***A***	***A***	***A***	***A***	M
Poliana Castel-Bran	M	B	***A***	B	B	B	M	***A***	M	B

A – Alta Intensidade; B – Baixa Intensidade; M – Média Intensidade;

Tabela 3. Resultados de Regressão Linear para Função de Complexidade.

Modelo	Resultado	Erro Residual
tempo (t) e vértices (n)	$\Theta(e^{0.03243}n^{0.39904})$	2.768
tempo (t) e arestas (m)	$\Theta(e^{3.71019}m^{-0.03846})$	2.811
tempo (t), vértices (n) e arestas (m)	$\Theta(e^{-6,334}n^{6,918}m^{-3,044})$	1.637

Referências

Batsyn, M. et al. (2013). "Improvements to MCS algorithm for the maximum clique problem". In *Journal of Combinatorial Optimization*. Springer, 2013. ISSN: 1382-6905.

Bomze, I. M. et al. (1999) "The Maximum Clique Problem". In Du, D. Z. e Pardalos, P. M. *Hankbook of combinatorial optimization*, Kluwer Academic Publishers.

Butenko, S. e Wilhelm, W. E. (2006) "Clique-detection models in computational biochemistry and genomics". In *European journal of operational research*, College Station.

Garey, M. R. e Johnson, D. S. (1979) "Computers and intractability: a guide to the theory of NP-Completeness", Bell Telephone Laboratories Inc. ISBN 0-7167-1044-7.

Grosso, A., Locatelli, M. e Pullan, W. (2007) "Simple ingredients leading to very efficient heuristics for the maximum clique problem". In *Journal of Heuristics*, Springer. ISSN: 1381-1231.

Mcclelland, D. C. (1972) "A sociedade competitiva: realização & progresso social", Rio de Janeiro: Expressão e Cultura.

Pessoa, M. C. (2011) "Ferramenta de monitoramento do comportamento empreendedor", Trabalho de Conclusão de Curso de Graduação em Ciência da Computação, UNIVALI, Itajaí.

Rese, A. L. R. e Santiago, R. (2012) "Ensino de teoria da complexidade", In: *Anais do XXXII Congresso da Sociedade Brasileira de Computação*, Curitiba.

Rosa, S. B. e Morales, S. A. (2010) "Perfil empreendedor: uma representação gráfica. Metodologia de medida quantitativa de competências empreendedoras", In *Revue internationale de psychosociologie*, Paris.

Sipser, M. (2007) "Introdução à teoria da computação", 2.ed. São Paulo, Thomson.