Projeto e avaliação de técnicas para emparelhamentos desconexos

Denilson M. V. Santos
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
denilsonmvs@ufmg.br, denilsonmvs@gmail.com

***Resumo*—Este artigo aborda o problema do emparelhamento desconexo, uma variação NP-Completa do clássico problema de emparelhamento em grafos, focando no refinamento e na otimização de modelos de Programação Linear Inteira (PLI) e Programação com Restrições (CP-SAT). Partindo de formulações base, uma série de otimizações foram sistematicamente implementadas e avaliadas, incluindo o ajuste de parâmetros (Big M), o enrijecimento de restrições (k = c) e a introdução de diferentes abordagens para quebra de simetria. Os resultados experimentais demonstram que, embora as otimizações no modelo de PLI tenham apresentado um impacto misto ou negativo, as mesmas estratégias no modelo CP-SAT, particularmente as restrições de quebra de simetria, levaram a um ganho de desempenho significativo. A formulação final com CP-SAT se estabeleceu como a abordagem mais robusta e eficiente, sendo capaz de resolver instâncias previamente intratáveis dentro do tempo limite.**

***Palavras-chave*—emparelhamento desconexo, otimização combinatória, programação linear inteira, SAT, grafos.**

I. Introdução

Emparelhamentos em grafos são estruturas combinatórias fundamentais e amplamente estudadas na ciência da computação. Um emparelhamento é formalmente definido como um subconjunto de arestas $M \subseteq E$ de um grafo $G = (V, E)$, onde nenhuma das arestas em M compartilha um vértice em comum. Este trabalho se aprofunda em uma variação específica conhecida como P-emparelhamento, na qual o subgrafo induzido pelos vértices das arestas do emparelhamento, $G[M]$, deve satisfazer uma determinada propriedade P.

O foco deste estudo é o problema do emparelhamento desconexo, onde a propriedade P requer que o número de componentes conexos no subgrafo induzido seja igual a uma constante c, ou seja, $\kappa(G[M]) = c$. A relevância teórica e prática deste problema é acentuada por sua complexidade computacional, tendo sido demonstrado que ele é NP-Completo para qualquer $c \geq 2$ [1]. Para emparelhamentos conexos, onde $c = 1$, é provado que uma solução ótima pode ser obtida em tempo polinomial [2].

Ao não buscar otimizar a quantidade de arestas no emparelhamento, o problema se torna muito similar ao *Induced Matching*, pois ambos se concentram na existência de uma subestrutura com propriedades específicas, em vez de sua cardinalidade máxima.

A similaridade se torna explícita quando notamos que o emparelhamento induzido (*Induced Matching*) é um caso particular do emparelhamento desconexo.

Formalmente:

- O problema de decisão para o emparelhamento desconexo pergunta: "Existe um emparelhamento M com $|M| = k$ tal que o subgrafo induzido $G[V(M)]$ tenha c componentes conexas?"
- O problema de decisão para o emparelhamento induzido pergunta: "Existe um emparelhamento induzido M com $|M| = k$?"

Pela definição de um emparelhamento induzido, o subgrafo $G[V(M)]$ consiste em exatamente k arestas isoladas. Cada aresta é uma componente conexa. Portanto, para qualquer emparelhamento induzido M, temos $\kappa(G[V(M)]) = |M|$.

Isso significa que a busca por um emparelhamento induzido de tamanho k é equivalente a resolver o problema do emparelhamento desconexo com os parâmetros $|M| = k$ e $c = k$.

II. Metodologia

A metodologia adotada neste trabalho seguiu uma abordagem iterativa, organizada em ciclos de criação, avaliação e refinamento dos modelos propostos. Ao final de cada ciclo, os resultados obtidos eram analisados para identificar pontos de melhoria e guiar os próximos ajustes nas formulações, em um processo de aprimoramento contínuo.

Para a implementação e execução dos experimentos, foram utilizadas ferramentas específicas para cada paradigma. Os modelos de Satisfatibilidade Booleana (SAT) foram desenvolvidos e executados com o auxílio do CP-SAT Solver, parte da biblioteca Google OR-Tools [3]. Para a implementação dos modelos de Programação Linear Inteira (PLI), embora o planejamento inicial previsse o uso do GNU Linear Programming Kit (GLPK) [4], testes comparativos demonstraram que o solver CBC (COIN-OR Branch and Cut) [5] apresentava desempenho superior. Diante disso, optou-se pela substituição do GLPK pelo CBC, que foi utilizado através da biblioteca PuLP [6] de python, na condução dos experimentos. A fim de garantir uma comparação justa e direta com os resultados anteriores, todos os testes foram realizados utilizando o mesmo conjunto de instâncias do projeto do semestre anterior.

O núcleo da investigação consistiu na aplicação e avaliação de diferentes estratégias de otimização sobre as modelagens base, incluindo:

- Avaliação da adição de novas restrições às formulações existentes.
- Análise do impacto de modificações pontuais nas modelagens.
- Estudo de otimizações e parâmetros de configuração específicos aplicáveis aos solvers SAT e PLI.
- Investigação sobre a viabilidade de adaptar formulações de problemas similares para o contexto dos emparelhamentos desconexos.

O desempenho de cada modelo foi quantificado por meio de um conjunto de métricas bem definidas, permitindo uma análise comparativa rigorosa. As métricas avaliadas foram:

- O tempo médio e o respectivo desvio padrão para se encontrar uma solução ótima.
- O percentual de execuções que excederam o tempo limite estipulado (timeout).

Essa abordagem sistemática permitiu avaliar de forma controlada o impacto de cada modificação, visando alcançar o objetivo final de obter modelos com desempenho superior aos da pesquisa anterior.

A. Casos de Teste e Ambiente Experimental

Para a avaliação empírica das modelagens, foi utilizado um conjunto de instâncias geradas aleatoriamente, sendo as mesmas instâncias do projeto anterior, cujas características são detalhadas a seguir.

- **Tamanho dos Grafos:** Foram gerados grafos com 16 e 32 vértices ($n \in \{16, 32\}$).
- **Densidade de Arestas:** A densidade dos grafos variou no intervalo de 0 a 1, com incrementos de 0,0625. Para cada valor de densidade, foram geradas 16 instâncias distintas a fim de garantir a robustez estatística dos resultados.
- **Número de Componentes (c):** O número de componentes conexos a serem encontrados foi definido da seguinte forma:
  - Para grafos de 16 vértices: $c \in \{2, 3, 4\}$.
  - Para grafos de 32 vértices: $c \in \{2, 3, 4, 8\}$.

Todos os experimentos foram conduzidos em um notebook pessoal equipado com um processador Ryzen 5 5500U. Para garantir uma comparação justa entre as abordagens, a execução de cada instância foi limitada a um tempo máximo de 4 segundos e ao uso de uma única thread de processamento.

As modelagens implementadas recebem como entrada uma matriz de adjacência **m**, o número de vértices **n** do grafo e o número de componentes desejado **c**.

B. Formulações para SAT e PLI

Para resolver o problema do emparelhamento desconexo, foram desenvolvidos modelos para PLI e SAT baseados em uma ideia conceitual de "componentes conexos germinados". A seguir, descreve-se a lógica geral da formulação.

1) Modelo Base de Componentes Conexos: A formulação principal se baseia em duas ideias centrais: garantir que o subconjunto de arestas forme um emparelhamento e, em seguida, contar quantos componentes conexos esse emparelhamento induz.

- **Variáveis de Emparelhamento:** Para cada par de vértices (i, j) com uma aresta no grafo de entrada ($M_{i,j} = 1$), uma variável booleana $E_{i,j}$ é definida. Se $E_{i,j} = 1$, a aresta (i, j) pertence ao emparelhamento. A restrição fundamental de um emparelhamento é imposta: se uma aresta (i, j) é escolhida, nenhuma outra aresta que incide em i ou j pode ser selecionada. Matematicamente:

$$E_{i,j} \rightarrow (\forall k \neq i, j : E_{i,k} = 0 \land E_{j,k} = 0)$$

- **Contagem de Componentes:** Para contar os componentes, foi introduzido um conceito de "semente". Cada componente conexo no emparelhamento é representado por um único vértice "semente" que "germina".
  - Uma variável S_i indica se o vértice i é a semente de seu componente.
  - Uma variável G_i armazena o identificador do grupo (ou componente) ao qual o vértice i pertence.
  - Se um vértice i é uma semente ($S_i = 1$), seu grupo é ele mesmo ($G_i = i$).
  - A identidade do grupo é propagada através das arestas do emparelhamento: se $E_{i,j} = 1$, então os vértices i e j devem pertencer ao mesmo grupo ($G_i = G_j$).
  - O número total de componentes é simplesmente a soma de todas as sementes que germinaram. Esta soma deve ser igual ao parâmetro de entrada c.

C. Programação Linear Inteira: Modelo inicial

a) Variáveis:

- $E_{i,j}$: Variável binária que indica se a aresta (i, j) pertence ao emparelhamento.
- $Y_{i,j}$: Variável binária que indica se a aresta (i, j) está presente no subgrafo induzido $G[M]$.
- S_i: Variável binária que indica a germinação da "semente" i no vértice i.
- $G_{i,j}$: Variável binária que indica se o vértice i pertence ao grupo j.

b) Implementação das Restrições e Função Objetivo:

- **Restrições de Emparelhamento:** Um conjunto de restrições garante a validade do emparelhamento.
  - As matrizes de arestas são simétricas e sem laços:

$$\forall i, j : E_{i,j} = E_{j,i}, \quad Y_{i,j} = Y_{j,i}$$

$$\forall i : E_{i,i} = 0, \quad Y_{i,i} = 0$$

  - Somente arestas existentes no grafo original podem ser selecionadas:

$$\forall i, j : E_{i,j} \leq M_{i,j}$$

– No máximo uma aresta do emparelhamento pode incidir em cada vértice:

$$\forall i : \sum_{j=0}^{n-1} E_{i,j} \leq 1$$

- **Componentes e Grupos:** Restrições para contar e formar os componentes conexos.
  – O número de "sementes" (componentes) deve ser maior ou igual a c:

$$\sum_{i=0}^{n-1} S_i \geq c$$

  – Uma semente i germina se, e somente se, o vértice i pertence ao grupo i:

$$\forall i : S_i = G_{i,i}$$

  – Um vértice pertence a um grupo se, e somente se, pertence ao emparelhamento:

$$\forall i : \sum_{j=0}^{n-1} G_{i,j} \leq \sum_{j=0}^{n-1} E_{i,j}$$

  – **Arestas no Grafo Induzido:** A variável $Y_{i,j}$ é ativada se a aresta (i, j) existe no grafo original e ambos os vértices i e j fazem parte do emparelhamento. A porta lógica AND correspondente foi implementada com as seguintes restrições lineares:

$$Y_{i,j} \leq \sum_{k=0}^{n-1} E_{i,k} \quad , \quad Y_{i,j} \leq \sum_{k=0}^{n-1} E_{j,k}$$

$$Y_{i,j} \leq M_{i,j}$$

$$Y_{i,j} \geq M_{i,j} + \sum_{k=0}^{n-1} E_{i,k} + \sum_{k=0}^{n-1} E_{j,k} - 2$$

  – **Propagação de Grupo:** Se uma aresta $Y_{i,j}$ existe no grafo induzido, os vértices i e j devem pertencer ao mesmo grupo. A implicação lógica $Y_{i,j} \rightarrow (G_{i,k} = G_{j,k})$ foi linearizada da seguinte forma:

$$\forall i, j, k : \quad 1 - Y_{i,j} + G_{i,k} \geq G_{j,k}$$

$$\forall i, j, k : \quad 1 - Y_{i,j} + G_{j,k} \geq G_{i,k}$$

- **Função Objetivo:** O objetivo é maximizar a cardinalidade do emparelhamento:

$$\text{maximizar} \sum_{i<j} E_{i,j}$$

D. Modificação da representação de grupos

O segundo modelo de PLI representa uma refatoração significativa do modelo inicial, visando reduzir a complexidade e melhorar o desempenho. As principais alterações são detalhadas a seguir, contrastando as abordagens e apresentando as novas restrições.

- **Representação de Grupos:** A mudança mais impactante foi na forma de representar os grupos. O modelo inicial utiliza uma matriz de variáveis binárias $G_{i,j}$ para indicar se o vértice i pertence ao grupo j. O novo modelo substitui isso por uma única variável contínua por vértice, G_i, que armazena o identificador numérico do grupo, reduzindo as variáveis de grupo de n^2 para n. A atribuição do grupo a um vértice "semente" S_i é então linearizada com a técnica "Big M" da seguinte forma:

$$G_i - (i + 1) \leq (1 - S_i) \cdot M$$

$$(i + 1) - G_i \leq (1 - S_i) \cdot M$$

- **Eliminação de Variáveis Auxiliares:** A variável $Y_{i,j}$, que representava explicitamente uma aresta no grafo induzido no modelo inicial, foi completamente eliminada. Neste modelo, a lógica de propagação de grupo (se os vértices i e j estão conectados no emparelhamento, então $G_i = G_j$) é incorporada diretamente em duas restrições que usam "Big M", sem a necessidade de uma variável intermediária:

$$G_i - G_j \leq \left(1 - M_{i,j} + 1 - \sum_k E_{i,k} + 1 - \sum_k E_{j,k}\right) \cdot M$$

$$G_j - G_i \leq \left(1 - M_{i,j} + 1 - \sum_k E_{i,k} + 1 - \sum_k E_{j,k}\right) \cdot M$$

- **Complexidade das Restrições:** O modelo inicial possui uma complexidade de restrições de $O(n^3)$. Isso ocorre principalmente pela regra de propagação de grupo, que itera sobre todos os possíveis semeadores k para cada aresta (i, j). Neste modelo, ao eliminar essa iteração tripla, consegue reduzir a complexidade total das restrições para $O(n^2)$, tornando a formulação muito mais enxuta para o solver. As restrições de propagação mostradas no item anterior, por exemplo, iteram apenas sobre i e j.
- **Técnica de Linearização:** Enquanto o primeiro modelo usa uma combinação de restrições, o segundo modelo unifica a linearização de todas as regras conceituais através da técnica "Big M". Além da atribuição de semente e propagação de grupo, isso também é usado para garantir que um vértice fora do emparelhamento não pertença a nenhum grupo, o que é feito com a seguinte restrição:

$$G_i \leq \left(\sum_k E_{i,k}\right) \cdot M$$

E. Otimização do Parâmetro Big-M no Modelo PLI

Uma das otimizações investigadas no modelo de PLI foi o ajuste do valor da constante "Big M". Modelos de Programação Linear Inteira podem ser sensíveis a constantes numéricas muito grandes, que podem levar a uma perda de desempenho do solver. A formulação original utilizava um valor arbitrariamente alto para M. Após uma análise teórica das restrições, concluiu-se que um valor de $M = n$, onde n é o número de vértices do grafo, seria um limite superior válido e consideravelmente mais "apertado" (tight).

F. Modelo CP-SAT

O modelo inicial para CP-SAT foi caracterizado por um número elevado de variáveis.

a) Variáveis:

- $E_{i,j}$: Variável binária que indica se a aresta (i, j) pertence ao emparelhamento.
- H_i: Variável binária que indica se o vértice i pertence ao emparelhamento (i.e., tem aresta incidente).
- $Y_{i,j}$: Variável binária que indica se a aresta (i, j) está presente no grafo induzido $G[M]$.
- $S_{i,j}$: Variável binária que indica a germinação da "semente" i no vértice j.
- $Germinated_i$: Variável binária que indica se a "semente" i germinou (em qualquer vértice).
- Z: Variável inteira que conta o número total de "sementes" que germinaram.
- G_i: Variável inteira que armazena o identificador do grupo ao qual o vértice i pertence.

b) Restrições e Função Objetivo: O comportamento do modelo é regido por um conjunto de restrições lógicas.

- **Restrições de Emparelhamento e Grafo Induzido:**
  - As arestas são não direcionadas e não há laços.
  - Somente arestas que existem no grafo original podem ser selecionadas:

$$\forall i, j : \neg M_{i,j} \rightarrow \neg E_{i,j}$$

  - Cada vértice pode ter no máximo uma aresta no emparelhamento:

$$\forall i, j, k \text{ com } k \neq i \text{ e } k \neq j : \\ (E_{i,j} \rightarrow \neg E_{i,k}) \wedge (E_{i,j} \rightarrow \neg E_{k,j})$$

  - Um vértice H_i pertence ao emparelhamento se a soma de suas arestas incidentes for 1:

$$\forall i : H_i = \sum_{j=0}^{n-1} E_{i,j}$$

  - Uma aresta (i, j) existe no grafo induzido $G[M]$ se ela existe no grafo original e ambos os seus vértices estão no emparelhamento:

$$\forall i, j : Y_{i,j} \leftrightarrow (M_{i,j} \wedge H_i \wedge H_j)$$

- **Restrições de Componentes, Sementes e Grupos:**
  - A variável $Germinated_i$ é verdadeira se a semente i germinou em algum vértice j:

$$\forall i : Germinated_i = \sum_{j=0}^{n-1} S_{i,j}$$

  - Uma semente pode germinar no máximo uma vez:

$$\forall i : Germinated_i \leq 1$$

  - A variável Z conta o número total de sementes que germinaram e deve ser maior ou igual a c:

$$Z = \sum_{i=0}^{n-1} Germinated_i \quad , \quad Z \geq c$$

  - Se um vértice não está no emparelhamento, ele não pertence a nenhum grupo (identificado como -1):

$$\forall i : \neg H_i \rightarrow (G_i = -1)$$

  - Se a semente i germina no vértice j, então o vértice i pertence ao grupo j:

$$\forall i, j : S_{i,j} \rightarrow (G_i = j)$$

  - Se uma aresta (i, j) está no grafo induzido, seus vértices devem pertencer ao mesmo grupo:

$$\forall i, j : Y_{i,j} \rightarrow (G_i = G_j)$$

- **Função Objetivo:** O objetivo é maximizar a cardinalidade do emparelhamento:

$$\text{maximizar } \sum_{i<j} E_{i,j}$$

G. Alterações para o segundo modelo CP-SAT

O segundo modelo de CP-SAT foi uma evolução direta do primeiro, com o objetivo principal de reduzir o número de variáveis e simplificar a lógica de "germinação de sementes". Isso foi alcançado através das seguintes alterações:

- **Simplificação da Germinação de Sementes:** A alteração mais significativa foi a remoção da variável matricial $S_{i,j}$, que indicava se a "semente i" germinava no "vértice j". Essa variável foi substituída por uma única variável binária por vértice, S_i, que indica que a semente de um componente germinou no próprio vértice i. Essa mudança eliminou a necessidade da variável auxiliar $Germinated_i$, que contava se uma semente havia germinado.
- **Contagem Direta de Componentes:** Como consequência da simplificação anterior, a forma de contar os componentes foi alterada.
  - No **modelo inicial**, a contagem era feita indiretamente, somando as variáveis $Germinated_i$:

$$Z = \sum_{i=0}^{n-1} Germinated_i$$

  - No **segundo modelo**, a contagem é feita diretamente sobre as novas variáveis de semente S_i:

$$Z = \sum_{i=0}^{n-1} S_i$$

- **Atribuição de Grupo Semente:** A regra para definir a qual grupo um vértice pertence, caso ele seja uma semente, foi simplificada.
  - O **modelo inicial** tinha uma regra complexa que dependia de onde a semente germinava:

$$\forall i, j : S_{i,j} \rightarrow (G_i = j)$$

  - O **segundo modelo** adota uma regra muito mais direta, onde o grupo de um vértice semente é o seu próprio índice:

$$\forall i : S_i \rightarrow (G_i = i)$$

- **Verificação de Vértices no Emparelhamento:** Houve uma sutil, mas importante, alteração na forma de definir a variável H_i, que indica se um vértice pertence ao emparelhamento.
  - O **modelo inicial** usava uma soma aritmética:

$$\forall i : H_i = \sum_{j=0}^{n-1} E_{i,j}$$

  - O **segundo modelo** passou a usar uma disjunção lógica (OR), que é uma operação mais natural para um solver SAT:

$$\forall i : H_i \leftrightarrow \bigvee_{j=0}^{n-1} E_{i,j}$$

H. Terceiro modelo CP-SAT

O terceiro modelo de CP-SAT divergiu significativamente do segundo, introduzindo uma nova forma de representar e propagar a noção de "grupos" ou componentes conexos.

- **Mudança na Representação de Grupos:** Esta é a alteração central. O segundo modelo usava uma variável inteira G_i para armazenar o identificador do grupo do vértice i. O terceiro modelo substitui isso por uma matriz de variáveis booleanas $G_{i,j}$, onde $G_{i,j}$ é verdadeiro se o vértice i pertence ao componente cuja "semente" é o vértice j. Para garantir a consistência desta nova representação, uma restrição de exclusividade foi adicionada, assegurando que um vértice só pode pertencer a um único grupo:

$$\forall i, j, k \text{ com } j \neq k : G_{i,j} \rightarrow \neg G_{i,k}$$

- **Nova Definição de Semente e Contagem:** O conceito de uma variável de semente explícita (S_i) foi removido.
  - No **segundo modelo**, uma variável S_i era usada para indicar uma semente , e a contagem de componentes era $\sum S_i$.
  - No **terceiro modelo**, um vértice i é definido como a semente de seu componente se ele pertence ao seu próprio grupo ($G_{i,i}$ é verdadeiro). A contagem de componentes, Z, é, portanto, a soma da diagonal principal da matriz de grupos. Além disso, uma semente só pode germinar se o vértice de fato pertencer ao emparelhamento (H_i ser verdadeiro).

$$Z = \sum_{i=0}^{n-1} G_{i,i}$$

$$\forall i : \neg H_i \rightarrow \neg G_{i,i}$$

- **Eliminação da Aresta Induzida e Nova Propagação de Grupo:** O terceiro modelo eliminou completamente a variável auxiliar $Y_{i,j}$, que representava uma aresta no grafo induzido. A lógica de propagação de grupo, que antes dependia de $Y_{i,j}$, foi substituída por uma restrição lógica mais complexa e integrada.
  - O **segundo modelo** usava uma regra simples: se a aresta (i, j) está no grafo induzido ($Y_{i,j}$), os vértices pertencem ao mesmo grupo:

$$\forall i, j : Y_{i,j} \rightarrow (G_i = G_j)$$

  - O **terceiro modelo** implementa uma equivalência: se os vértices i e j estão conectados no grafo induzido, eles devem pertencer ao mesmo conjunto de grupos para todos os possíveis semeadores k. Isso é modelado pela seguinte restrição:

$$\forall i, j, k : (M_{i,j} \wedge H_i \wedge H_j) \rightarrow (G_{i,k} \leftrightarrow G_{j,k})$$

I. Enrijecimento da Restrição de Componentes

Outra modificação avaliada foi o enrijecimento da restrição sobre o número de componentes, que nos modelos base era de desigualdade ($\sum S_i \geq c$). Esta restrição foi alterada para uma de igualdade estrita ($\sum S_i = c$), que define o problema de forma mais precisa e tem o potencial de reduzir o espaço de busca do solver.

J. Otimização do Modelo PLI: Quebra de Simetrias

Uma otimização adicional foi explorada na formulação de PLI com o objetivo de reduzir o espaço de busca do solver por meio da quebra de simetrias. A principal fonte de simetria no modelo de componentes conexos reside na escolha arbitrária do vértice "semente". Para um mesmo componente no emparelhamento, qualquer um de seus vértices poderia ser escolhido como a semente, gerando múltiplas representações equivalentes da mesma solução.

Para eliminar essa redundância, foi introduzida uma regra de ordenamento: para qualquer aresta (i, j) no emparelhamento, com $i < j$, apenas o vértice de menor índice (i) tem permissão para ser a semente daquele componente. Esta regra é imposta pela seguinte restrição linear, adicionada ao modelo para todos os pares de vértices com $i < j$:

$$S_j \leq (1 - M_{i,j}) + \left(1 - \sum_k E_{i,k}\right) + \left(1 - \sum_k E_{j,k}\right)$$

Nesta inequação, sendo $M_{i,j}$ a matriz de adjacência do grafo de entrada, o lado direito da expressão torna-se zero se, e somente se, os vértices i e j estiverem ambos no emparelhamento e conectados por uma aresta. Sob estas condições, a restrição força $S_j = 0$, proibindo que o vértice de maior índice na aresta seja a semente e quebrando efetivamente a simetria. Em todos os outros casos, o lado direito é maior ou igual a 1, tornando a restrição trivial para a variável binária S_j.

Para este teste final, a restrição de quebra de simetria foi adicionada ao modelo de PLI que já continha as duas otimizações anteriores: o ajuste do Big M ($M = n$) e a restrição de igualdade para o número de componentes ($k = c$).

Foi tentada uma segunda abordagem que testou uma regra de ordenamento mais simples e restritiva, aplicada seletivamente para cada aresta (i, j) do grafo original com $i < j$:

$$S_j \leq 1 - \sum_k E_{i,k}$$

A regra impõe que, se o vértice de menor índice i pertencer ao emparelhamento, o vértice de maior índice j (conectado a i) não pode ser uma semente.

K. Otimização do Modelo CP-SAT: Quebra de Simetrias

De forma análoga ao modelo PLI, foram exploradas estratégias de quebra de simetria para a formulação em CP-SAT. O objetivo é o mesmo: eliminar representações de soluções equivalentes, que surgem da escolha arbitrária do vértice "semente" para cada componente, e assim guiar o solver para uma convergência mais rápida. Duas abordagens baseadas em regras de ordenamento foram testadas.

1) Abordagem 1: Restrição por Par de Vértices no Emparelhamento: A primeira abordagem impõe uma regra que depende da participação de ambos os vértices de uma aresta no emparelhamento. A regra lógica é: se uma aresta (i, j) do grafo original com $j \leq i$ tem ambos os seus vértices no emparelhamento, então o vértice de menor ou igual índice, j, não pode ser uma semente.

Isso é modelado pela seguinte cláusula booleana, que é equivalente a uma implicação:

$$(H_i \wedge H_j) \rightarrow \neg G_{j,j} \quad (\forall (i, j) \in E \text{ com } j \leq i)$$

onde H_i é verdadeiro se o vértice i está no emparelhamento, $G_{j,j}$ é verdadeiro se j é uma semente, e E é o conjunto de arestas do grafo de entrada.

2) Abordagem 2: Restrição Simplificada: A segunda abordagem testou uma regra de ordenamento mais simples e agressiva. A condição para proibir um vértice de ser semente depende apenas da participação do seu vizinho de maior índice no emparelhamento. A regra é: se uma aresta (i, j) existe no grafo com $j \leq i$, e o vértice i está no emparelhamento, então o vértice j não pode ser uma semente.

A formulação lógica desta regra é a seguinte:

$$H_i \rightarrow \neg G_{j,j} \quad (\forall (i, j) \in E \text{ com } j \leq i)$$

L. Backtrack Simples

Foi testada uma abordagem de backtrack simples, projetada para explorar sistematicamente o espaço de soluções. A lógica do algoritmo se baseia em uma chamada recursiva que, para cada vértice do grafo, explora dois ramos: um em que o vértice é mantido com suas arestas e outro em que todas as suas arestas são removidas, efetivamente o isolando de possíveis emparelhamentos naquele subproblema.

A cada passo da recursão, o algoritmo primeiro verifica se o subgrafo atual G já satisfaz a condição de ter pelo menos c componentes conexos. Em caso afirmativo, ele encontrou um subgrafo válido e pode computar a solução ótima para ele usando o algoritmo de Blossom, que encontra o emparelhamento máximo em tempo polinomial. Caso contrário, a busca continua explorando as duas ramificações.

A complexidade desta abordagem é de $O(2^n \cdot n^3)$, combinando a exploração exponencial do espaço de busca com a complexidade cúbica do algoritmo de Blossom, que pode ser chamado em cada folha da árvore de recursão.

M. Otimização com Branch and Bound

Para mitigar as deficiências do backtrack simples, foi desenvolvida uma versão com a técnica de Branch and Bound. O objetivo era podar ramos da árvore de busca que não poderiam levar a uma solução ótima, reduzindo o espaço de busca explorado.

1) Heurística e Poda de Ramos: A heurística adotada foi o valor retornado pelo algoritmo de Blossom, usado como um limitante superior (upper bound) para o tamanho do emparelhamento em um determinado ramo da busca. Adicionalmente, foi desenvolvido um algoritmo de poda específico para lidar com grafos densos, que tendem a não possuir soluções válidas. Este algoritmo, com complexidade $O(nm)$, verifica se a remoção de um vértice tem o potencial de aumentar o número de componentes conexos do grafo, descartando prematuramente estados que não podem gerar soluções.

A expansão de cada estado na busca do Branch and Bound tinha, portanto, uma complexidade de $O(n^3 + mn)$, somando o custo da heurística (Blossom) e da verificação de poda.

III. Resultados

O desempenho de cada abordagem e de suas otimizações foi medido pelo tempo médio de execução e pela taxa de timeouts. Os resultados detalhados são apresentados nos gráficos do Apêndice A.

A. Resultados das otimizações no modelo PLI

- **Modelo inicial** Figuras 1 a 7
- **Modelo com modificação da representação de grupos** Figuras 36 a 42
- **Ajuste do Big-M:** Figuras 50 a 56.
- **Restrição k = c:** Figuras 57 a 63.
- **Quebra de simetria:** Figuras 71 a 77.
- **Quebra de simetria simplificada** A restrição simplificada revelou uma falha crítica. Foi observado que, para um caso de teste que sabidamente não possuía solução

viável, o solver encontrou uma solução quando esta restrição foi aplicada. Uma análise posterior confirmou que a solução encontrada era inválida, violando as premissas do problema. A hipótese para este comportamento é que a simplificação da restrição introduziu uma fragilidade na formulação, levando a uma instabilidade numérica que resultou em uma resposta incorreta do solver. Devido a esta falha, esta abordagem foi descartada.

B. Resultados das otimizações no modelo CP-SAT

- **Modelo inicial** Figuras 8 a 14
- **Segundo modelo** Figuras 15 a 21
- **Terceiro modelo** Figuras 43 a 49
- **Restrição k = c:** Figuras 64 a 70.
- **Quebra de simetria:** Abordagem 1: Figuras 78 a 84. Abordagem 2: Figuras 85 a 91.

C. Resultados backtrack e branch and bound

- **Backtrack simples** Figuras 22 a 28
- **Branch and bound** Figuras 29 a 35

IV. Discussão

Nota: As análises de desempenho nesta seção são sempre feitas em comparação com o modelo imediatamente anterior na sequência de desenvolvimento, salvo indicação em contrário.

A. Análise dos Modelos Iniciais e Evolução

A primeira fase da pesquisa focou em desenvolver e refinar modelos base de PLI e CP-SAT, com as seguintes observações de desempenho:

a) Evolução do Modelo PLI: O modelo de PLI inicial, com sua complexidade de restrições de $O(n^3)$, mostrou-se inviável na prática. Ele atingiu o tempo limite em quase todas as instâncias, mesmo para grafos pequenos de 16 vértices, não conseguindo obter resultados satisfatórios. A refatoração para o segundo modelo, que reduziu a complexidade para $O(n^2)$ ao eliminar variáveis e unificar a lógica com a técnica "Big M", representou um avanço crucial. Este modelo reduzido conseguiu resolver a maioria dos casos para grafos de 16 vértices, mas ainda apresentava dificuldades com instâncias maiores de 32 vértices, onde o tempo de execução aumentava consideravelmente e os timeouts ainda eram frequentes. Isso estabeleceu que, embora a formulação fosse viável, ela ainda precisava de otimizações para escalar.

b) Evolução do Modelo CP-SAT: A progressão dos modelos CP-SAT demonstrou um ganho de performance ainda mais expressivo. O primeiro modelo apresentou resultados satisfatórios para grafos de 16 vértices, mas falhou completamente nos de 32 vértices, atingindo o tempo limite em quase todos os casos. A transição para o segundo modelo, que simplificou drasticamente a lógica de "germinação de sementes", foi um ponto de virada: ele não só resolveu todos os casos para 16 vértices com extrema rapidez, como também obteve sucesso em uma porção significativa dos grafos de 32 vértices. O terceiro modelo, que alterou a representação de grupos para uma matriz booleana, teve um desempenho misto: foi ligeiramente inferior ao segundo modelo para 16 vértices, mas ligeiramente superior para 32 vértices.

B. Análise das Abordagens de Backtracking

As abordagens algorítmicas, embora interessantes, mostraram-se menos competitivas que os modelos de programação matemática.

a) Backtrack Simples: Esta abordagem foi extremamente eficiente para os grafos de 16 vértices, superando os modelos de PLI e SAT iniciais em velocidade. No entanto, sua natureza exponencial ($O(2^n \cdot n^3)$) o tornou completamente inviável para as instâncias de 32 vértices, que invariavelmente excediam o tempo limite.

b) Branch and Bound: A versão com Branch and Bound foi desenvolvida para mitigar a falha de escalabilidade do backtrack simples. O custo da heurística e das verificações de poda tornou-o mais lento para os casos de 16 vértices. Contudo, para os grafos de 32 vértices, a poda de ramos foi eficaz, permitindo resolver algumas instâncias (especialmente as mais densas) que o backtrack simples não conseguia. Mesmo assim, seu desempenho geral foi inferior ao dos modelos CP-SAT refinados, justificando o foco nestes últimos para a fase de otimização final.

C. Análise da Otimização do Parâmetro Big-M

A primeira otimização investigada foi o ajuste do parâmetro "Big M" para um valor mais restrito ($M = n$). A expectativa era que um limite superior mais "apertado" (tight) pudesse melhorar a estabilidade numérica e guiar o solver para uma solução mais rapidamente.

Contudo, a análise comparativa dos resultados não revelou uma melhora de performance clara ou consistente. Ao comparar os resultados do modelo base com os da versão otimizada, nota-se um comportamento misto:

- Para as instâncias menores, com 16 vértices, a alteração resultou em uma sutil degradação do desempenho, com tempos de execução, em média, ligeiramente maiores.
- Para as instâncias maiores, com 32 vértices, o impacto foi ambíguo. Observou-se uma melhora marginal em alguns cenários e uma pequena piora em outros, não sendo possível apontar um benefício claro e generalizado.

Uma possível interpretação para este resultado é que, para a escala de problemas testada, o modelo não sofre de instabilidade numérica a ponto de o ajuste do "Big M" ser um fator crítico.

D. Análise do Enrijecimento da Restrição de Componentes

A alteração da restrição de desigualdade para uma de igualdade estrita (buscando exatamente c componentes) foi testada, com a hipótese de que um modelo mais restrito reduziria o espaço de busca e, consequentemente, o tempo de solução. Contudo, na prática, os resultados não confirmaram essa expectativa, mostrando um impacto diferente para cada modelo.

Para o modelo de PLI, a mudança não resultou em ganhos de performance. Para as instâncias de 16 vértices, observou-se uma sutil piora nos tempos médios de execução. Para 32 vértices, os resultados foram ambíguos, com pequenas variações que não indicam uma tendência clara de melhora.

O modelo CP-SAT mostrou-se em grande parte indiferente a esta alteração. Conforme os resultados, não houve uma diferença significativa de desempenho entre as duas formulações.

E. Análise das Otimizações por Quebra de Simetria

As estratégias de quebra de simetria foram as que apresentaram o impacto mais significativo e divergente entre os modelos PLI e CP-SAT.

Para o modelo de PLI, a introdução da restrição de quebra de simetria produziu resultados mistos. Para as instâncias menores, de 16 vértices, foi notada uma sutil melhora de desempenho, particularmente em grafos mais esparsos. No entanto, para as instâncias maiores, de 32 vértices, a mesma restrição causou uma degradação significativa na performance. A hipótese mais provável é que a sobrecarga computacional imposta por essas restrições adicionais superou o benefício obtido com a poda do espaço de busca.

Em contrapartida, para o modelo CP-SAT, ambas as estratégias de quebra de simetria resultaram em uma melhora de desempenho acentuada e inequívoca. Ambas as abordagens levaram a uma nítida redução no tempo médio de resolução e, mais importante, permitiram ao solver encontrar soluções ótimas para instâncias que antes resultavam em timeout. Ao comparar as duas estratégias, a abordagem com a restrição simplificada ($H_i \rightarrow \neg G_{j,j}$) apresentou uma ligeira vantagem, estabelecendo-se como a formulação de melhor desempenho.

Duas tendências foram notadas: o ganho de performance foi mais expressivo em grafos esparsos, e o impacto da otimização diminuiu à medida que o número de componentes (c) aumentou. Uma razão plausível é que, com um c maior, os componentes são forçados a ser menores, possuindo inerentemente menos simetrias e reduzindo o benefício marginal de adicionar restrições para esse fim.

V. Conclusão

Este trabalho apresentou uma investigação abrangente sobre técnicas para a resolução do problema do emparelhamento desconexo, partindo de uma exploração de diversas abordagens até o refinamento e otimização sistemática dos modelos mais promissores. O objetivo central foi identificar formulações robustas e eficientes para este problema NP-Completo, avaliando o impacto de diferentes estratégias de modelagem e otimização.

A principal conclusão é que, para a modelagem baseada em "sementes" e a escala de problemas testada, a abordagem com o solver CP-SAT mostrou-se consistentemente superior à de PLI. O desempenho do modelo CP-SAT, especialmente após as otimizações, foi notavelmente mais robusto e rápido.

A otimização de maior impacto investigada foi a quebra de simetrias, que, no entanto, produziu efeitos drasticamente diferentes em cada paradigma. No modelo CP-SAT, a adição de restrições de ordenamento levou a um ganho de desempenho significativo, permitindo resolver instâncias previamente intratáveis dentro do tempo limite. Em contraste, no modelo PLI, as mesmas estratégias resultaram em uma degradação de performance para os grafos de maior porte, evidenciando a alta sensibilidade da formulação linear à sobrecarga computacional imposta por este tipo de restrição. Outras modificações, como o ajuste do parâmetro "Big M" e a alteração da restrição de contagem de componentes para uma igualdade estrita, não apresentaram benefícios claros para nenhum dos modelos e, em alguns casos, levaram a instabilidades numéricas na formulação de PLI.

As abordagens algorítmicas baseadas em backtrack também foram investigadas na fase exploratória do trabalho. A versão simples foi eficiente para grafos de 16 vértices, mas sua complexidade exponencial a tornou inviável para instâncias maiores. Uma otimização com Branch and Bound melhorou a escalabilidade para alguns casos de 32 vértices, mas com um custo de overhead que a tornava mais lenta em cenários mais simples. No geral, essas abordagens se mostraram menos robustas que os modelos de programação matemática, justificando o foco subsequente em PLI e CP-SAT.

Em futuras pesquisas, o problema de *induced matching* poderia ser explorado como uma ferramenta eficaz para a detecção de casos sem solução em emparelhamentos desconexos devido à grande similaridade entre os dois problemas.

Referências

[1] G. C. M. Gomes, B. P. Masquio, P. E. D. Pinto, V. F. dos Santos, and J. L. Szwarcfiter, "Disconnected matchings," *Theoretical Computer Science*, 2023.

[2] S. Micali and V. V. Vazirani, "An O(v—v— c —E—) algorithm for finding maximum matching in general graphs," in *21st Annual Symposium on Foundations of Computer Science (SFCS 1980)*, 1980, pp. 17-27. doi: 10.1109/SFCS.1980.12.

[3] Google, "Google OR-Tools," 2025. [Online]. Available: https://developers.google.com/optimization

[4] A. O. Makhorin, *GLPK (GNU Linear Programming Kit), Version 5.0*, Free Software Foundation, 2020. [Online]. Available: https://www.gnu.org/software/glpk/

[5] COIN-OR Foundation, "COIN-OR Branch and Cut (CBC)," 2025. [Online]. Available: https://github.com/coin-or/Cbc

[6] S. Mitchell, M. O'Sullivan, and I. Dunning, "PuLP: A Linear Programming Toolkit for Python," in *Proc. 8th Python in Science Conf. (SciPy 2011)*, 2011.

Apêndice

- **Repositório no GitHub:** https://github.com/DenilsonMVS/Disconnected-Matchings
- **Pitch Parcial POC1:** https://youtu.be/WbX5ezcpXyw
- **Pitch Final: POC1:** https://youtu.be/3xWutbMYo6g
- **Pitch Parcial POC2:** https://youtu.be/kN_F_KFTGVs
- **Pitch Final: POC2:** https://youtu.be/PYspfjS70Lc

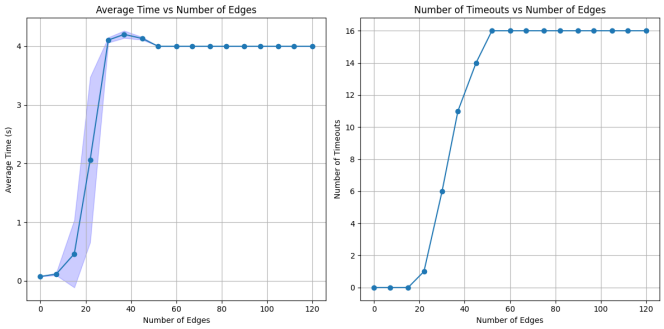

Figura 1. Resultados (PLI Inicial) para 16 vértices, c=2.

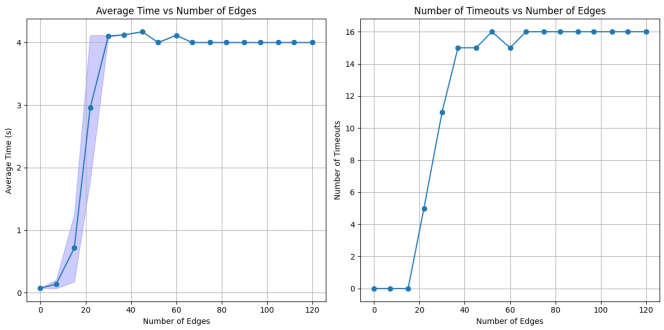

Figura 2. Resultados (PLI Inicial) para 16 vértices, c=3.

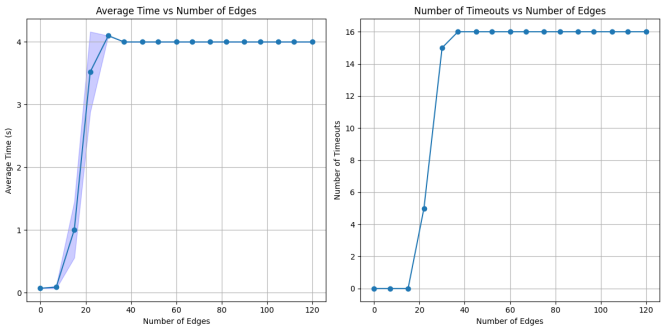

Figura 3. Resultados (PLI Inicial) para 16 vértices, c=4.

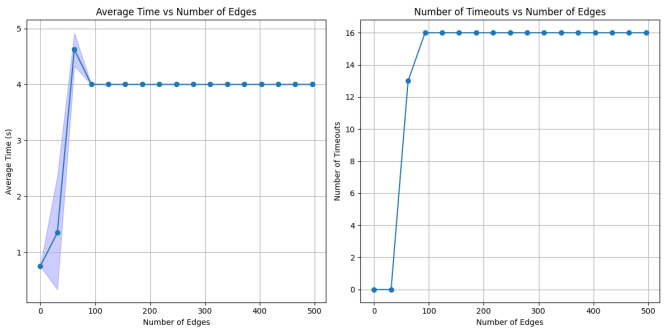

Figura 4. Resultados (PLI Inicial) para 32 vértices, c=2.

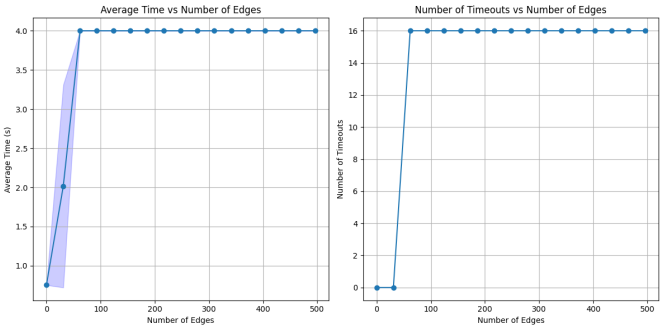

Figura 5. Resultados (PLI Inicial) para 32 vértices, c=3.

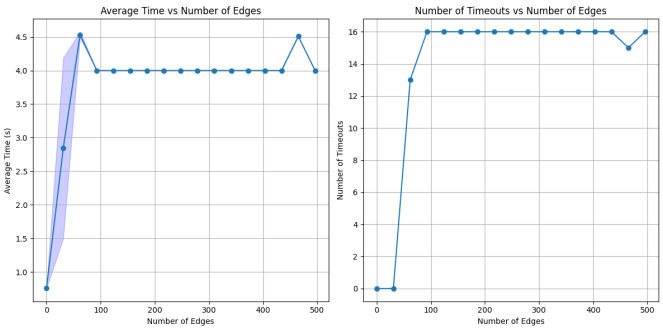

Figura 6. Resultados (PLI Inicial) para 32 vértices, c=4.

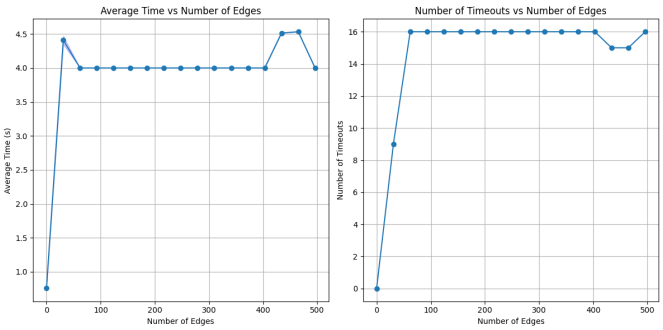

Figura 7. Resultados (PLI Inicial) para 32 vértices, c=8.

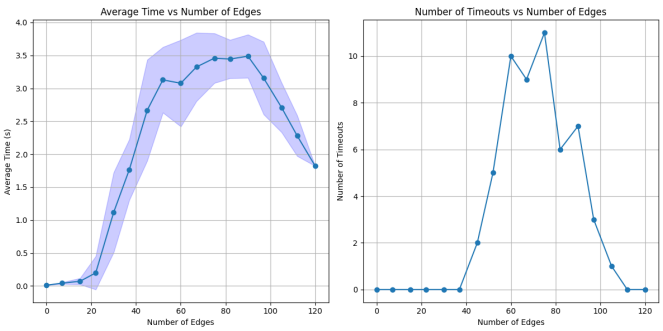

Figura 8. Resultados (CP-SAT Inicial) para 16 vértices, c=2.

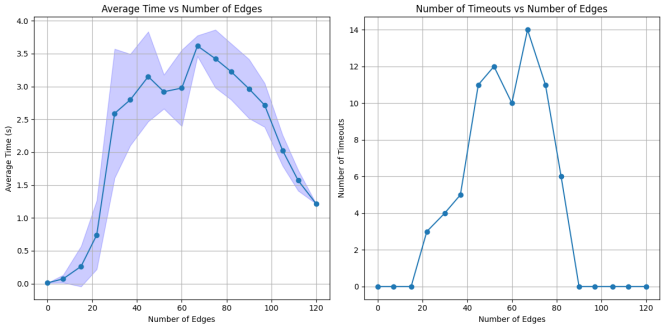

Figura 9. Resultados (CP-SAT Inicial) para 16 vértices, c=3.

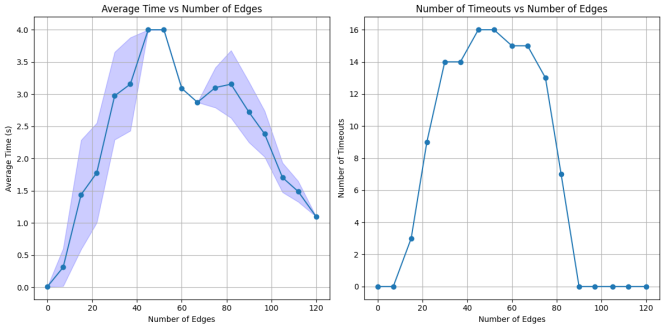

Figura 10. Resultados (CP-SAT Inicial) para 16 vértices, c=4.

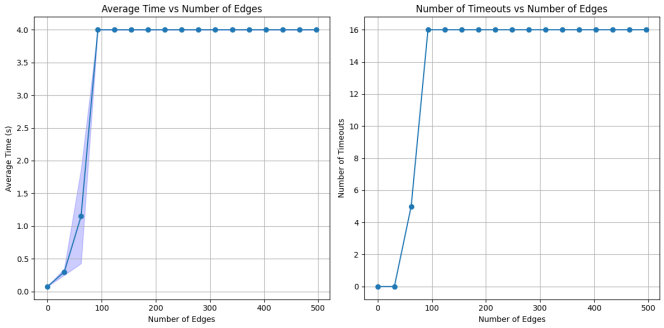

Figura 11. Resultados (CP-SAT Inicial) para 32 vértices, c=2.

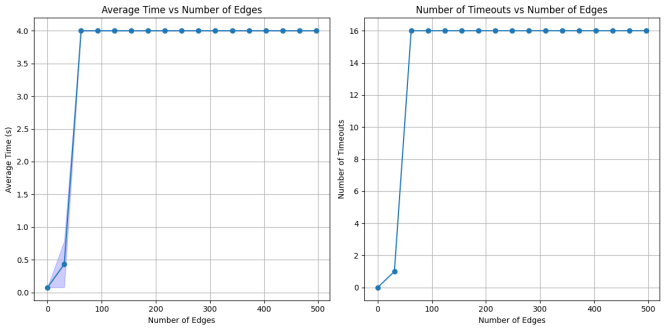

Figura 12. Resultados (CP-SAT Inicial) para 32 vértices, c=3.

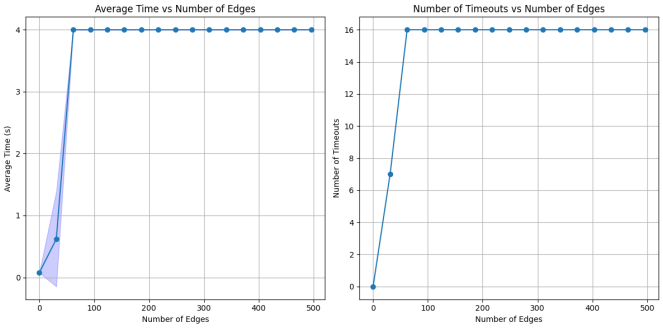

Figura 13. Resultados (CP-SAT Inicial) para 32 vértices, c=4.

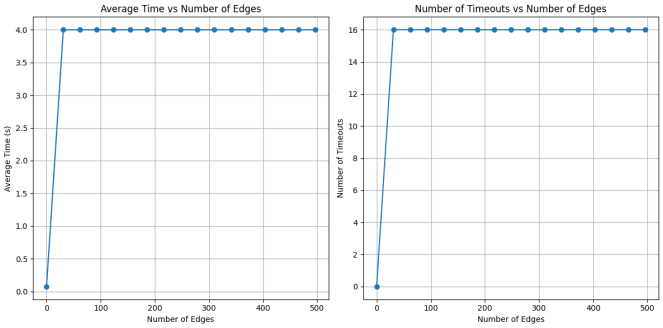

Figura 14. Resultados (CP-SAT Inicial) para 32 vértices, c=8.

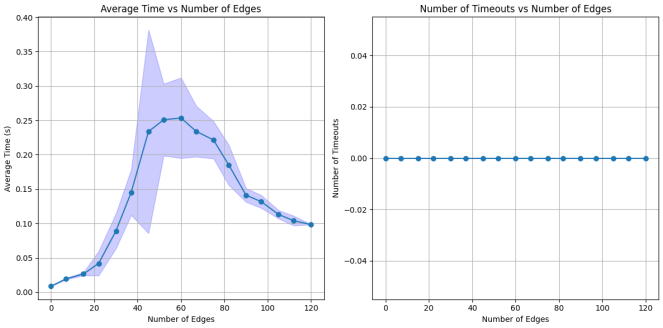

Figura 15. Resultados (CP-SAT Segundo Modelo) para 16 vértices, c=2.

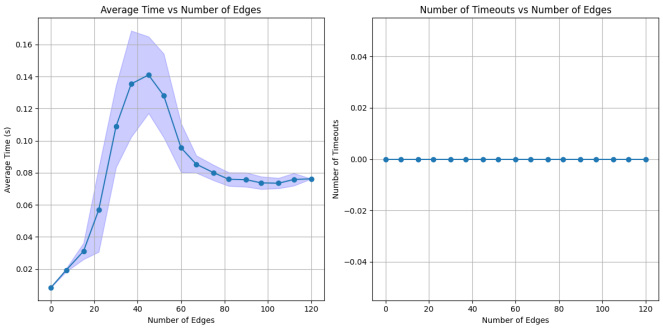

Figura 16. Resultados (CP-SAT Segundo Modelo) para 16 vértices, c=3.

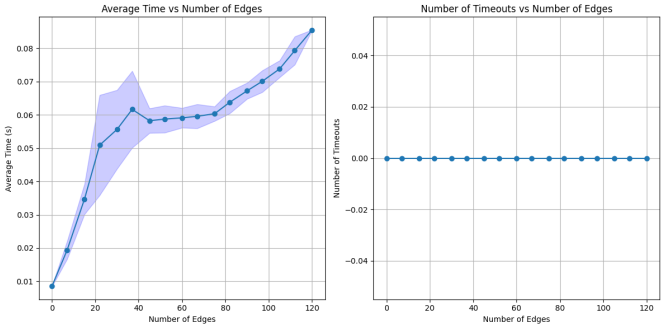

Figura 17. Resultados (CP-SAT Segundo Modelo) para 16 vértices, c=4.

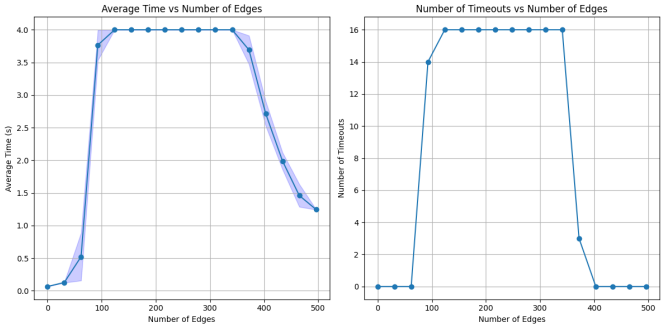

Figura 18. Resultados (CP-SAT Segundo Modelo) para 32 vértices, c=2.

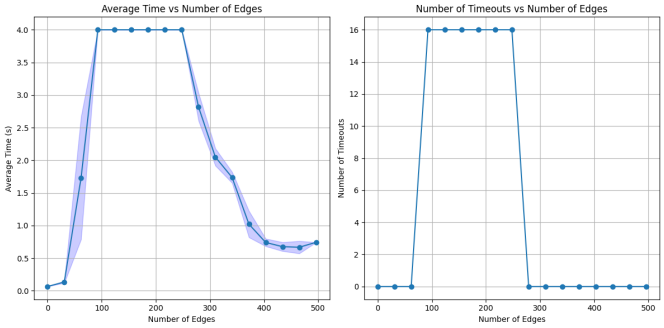

Figura 19. Resultados (CP-SAT Segundo Modelo) para 32 vértices, c=3.

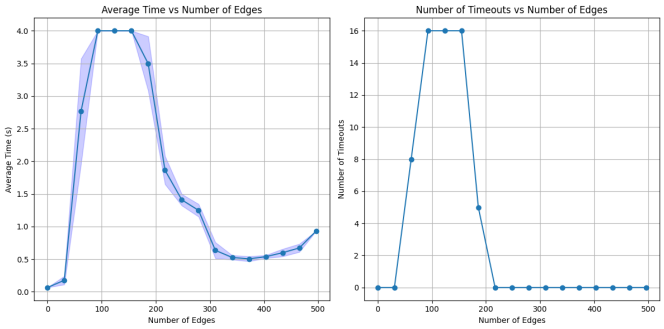

Figura 20. Resultados (CP-SAT Segundo Modelo) para 32 vértices, c=4.

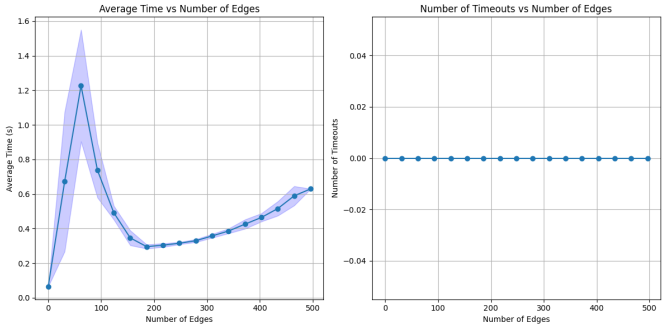

Figura 21. Resultados (CP-SAT Segundo Modelo) para 32 vértices, c=8.

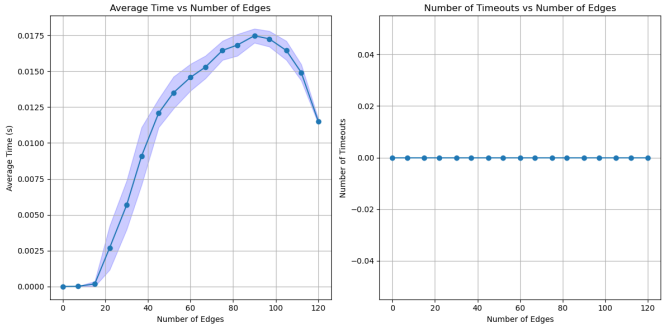

Figura 22. Resultados (Backtrack Simples) para 16 vértices, c=2.

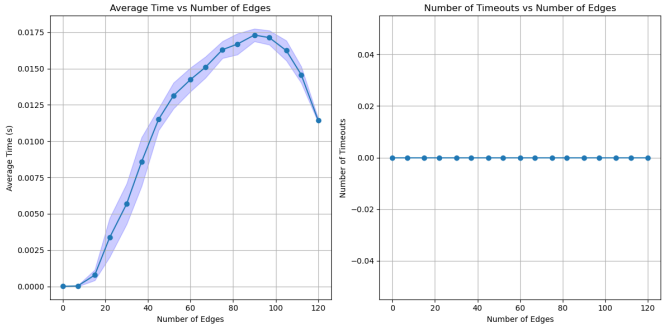

Figura 23. Resultados (Backtrack Simples) para 16 vértices, c=3.

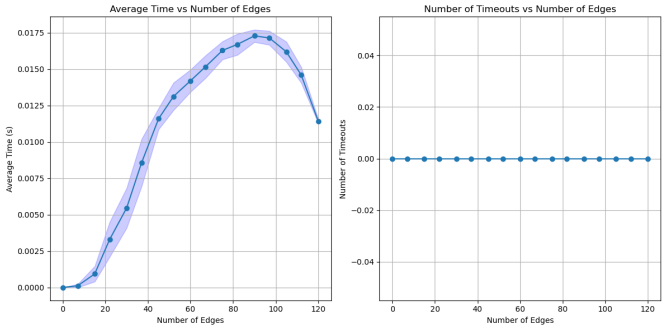

Figura 24. Resultados (Backtrack Simples) para 16 vértices, c=4.

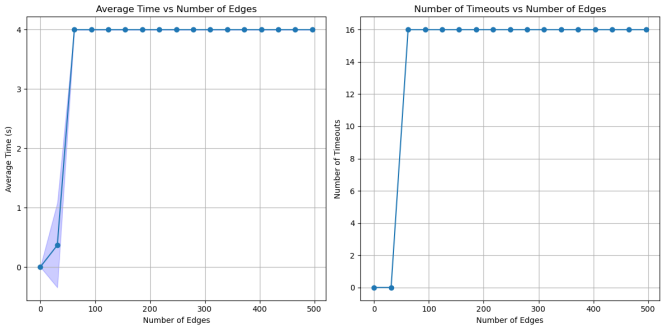

Figura 25. Resultados (Backtrack Simples) para 32 vértices, c=2.

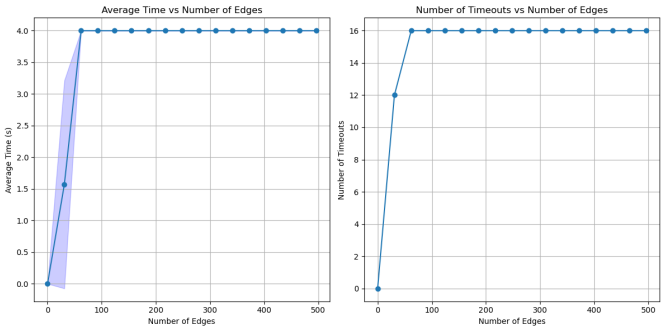

Figura 26. Resultados (Backtrack Simples) para 32 vértices, c=3.

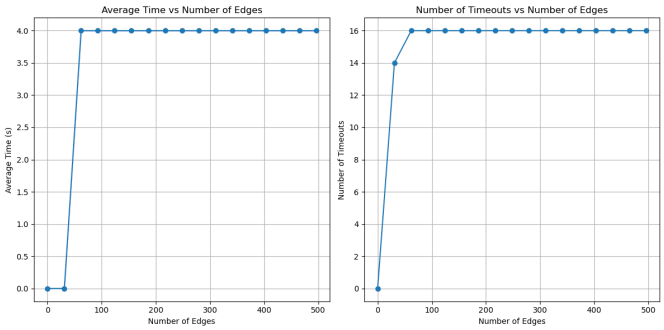

Figura 27. Resultados (Backtrack Simples) para 32 vértices, c=4.

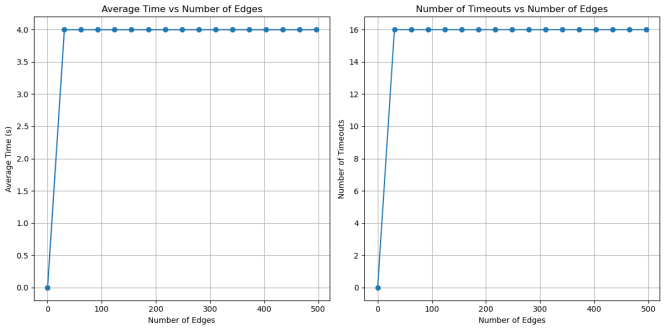

Figura 28. Resultados (Backtrack Simples) para 32 vértices, c=8.

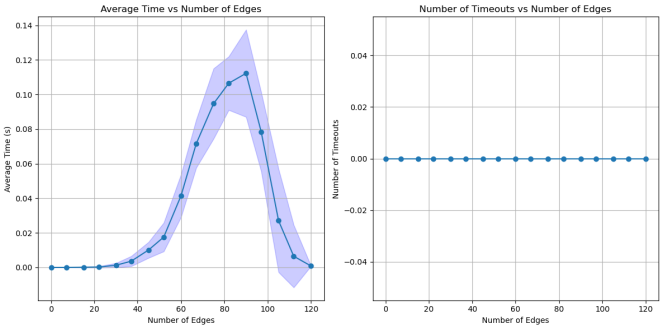

Figura 29. Resultados (Branch and Bound) para 16 vértices, c=2.

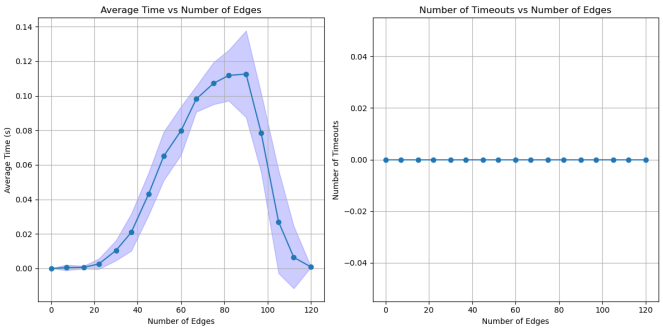

Figura 30. Resultados (Branch and Bound) para 16 vértices, c=3.

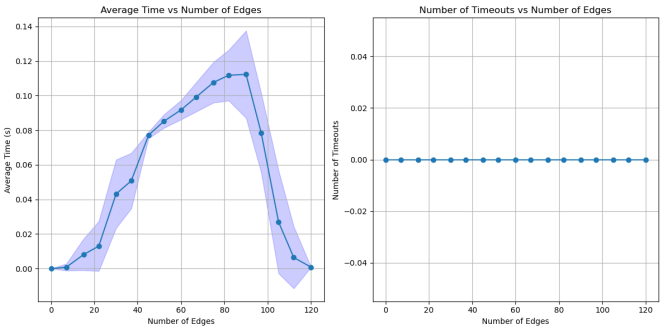

Figura 31. Resultados (Branch and Bound) para 16 vértices, c=4.

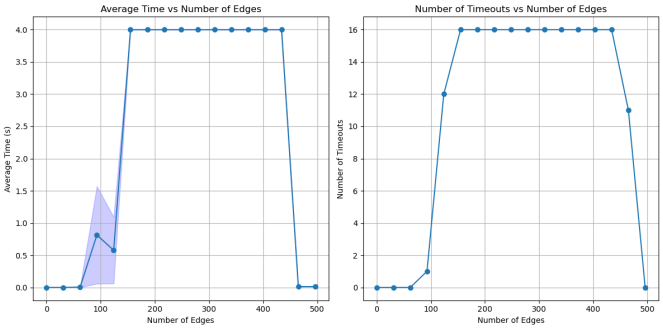

Figura 32. Resultados (Branch and Bound) para 32 vértices, c=2.

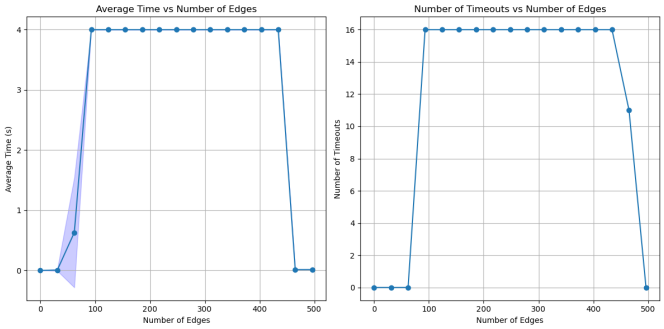

Figura 33. Resultados (Branch and Bound) para 32 vértices, c=3.

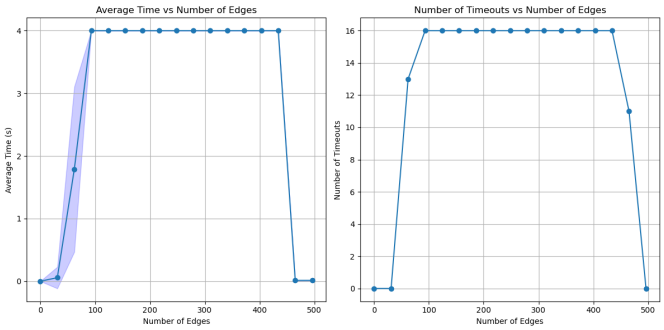

Figura 34. Resultados (Branch and Bound) para 32 vértices, c=4.

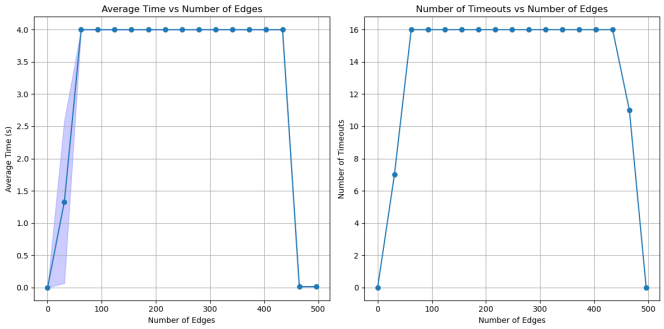

Figura 35. Resultados (Branch and Bound) para 32 vértices, c=8.

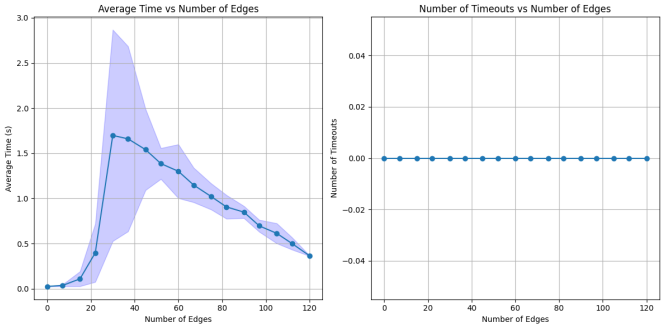

Figura 36. Gráfico de resultados para instâncias com 16 vértices e objetivo de 2 componentes conexos.

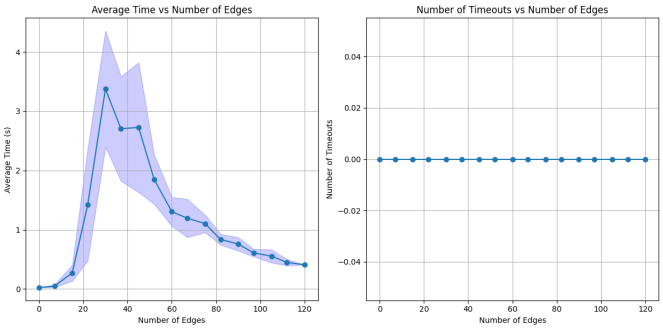

Figura 37. Gráfico de resultados para instâncias com 16 vértices e objetivo de 3 componentes conexos.

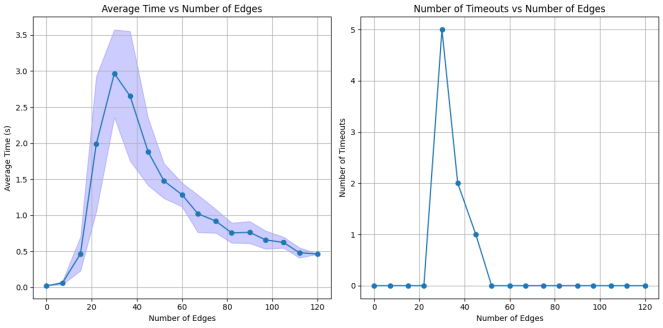

Figura 38. Gráfico de resultados para instâncias com 16 vértices e objetivo de 4 componentes conexos.

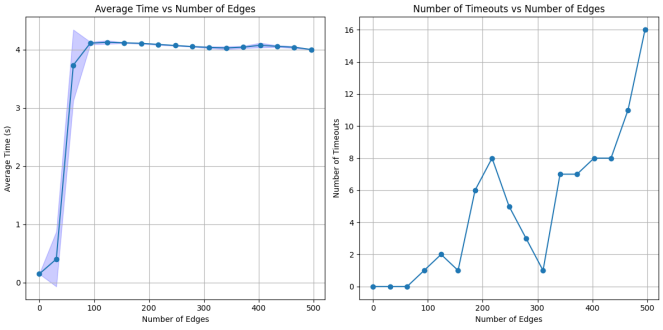

Figura 39. Gráfico de resultados para instâncias com 32 vértices e objetivo de 2 componentes conexos.

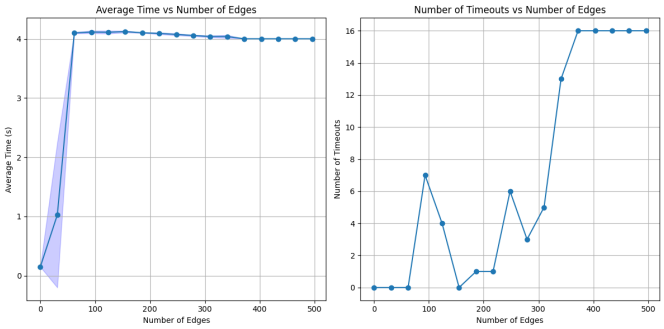

Figura 40. Gráfico de resultados para instâncias com 32 vértices e objetivo de 3 componentes conexos.

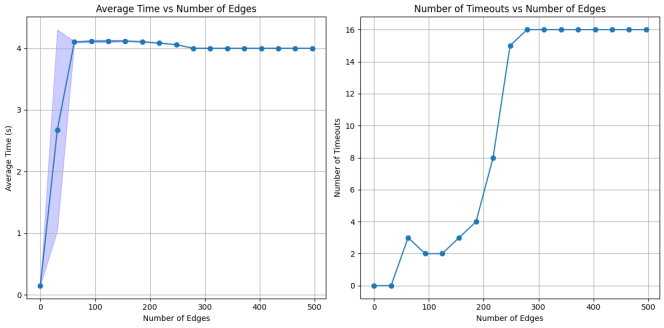

Figura 41. Gráfico de resultados para instâncias com 32 vértices e objetivo de 4 componentes conexos.

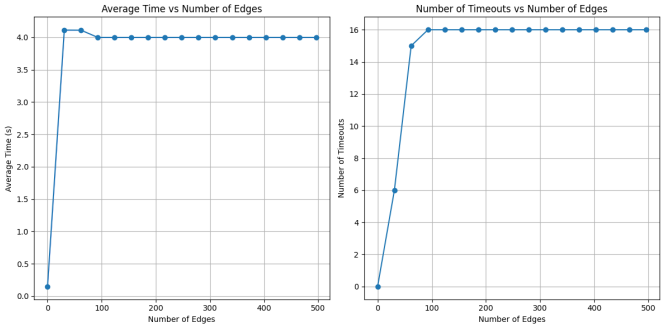

Figura 42. Gráfico de resultados para instâncias com 32 vértices e objetivo de 8 componentes conexos.

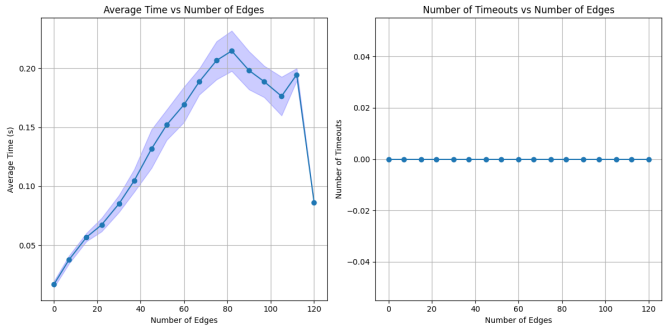

Figura 43. Gráfico de resultados (CP-SAT) para instâncias com 16 vértices e objetivo de 2 componentes conexos.

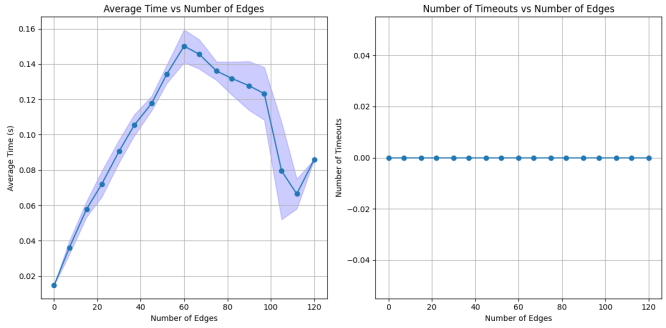

Figura 44. Gráfico de resultados (CP-SAT) para instâncias com 16 vértices e objetivo de 3 componentes conexos.

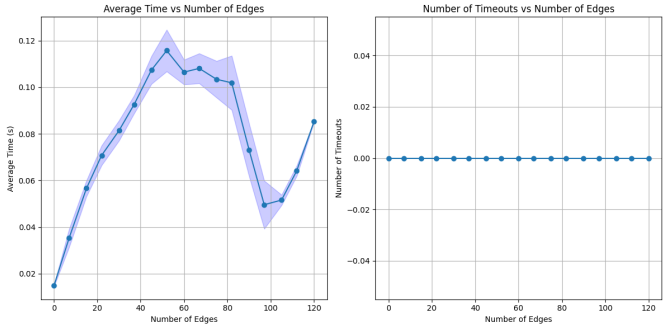

Figura 45. Gráfico de resultados (CP-SAT) para instâncias com 16 vértices e objetivo de 4 componentes conexos.

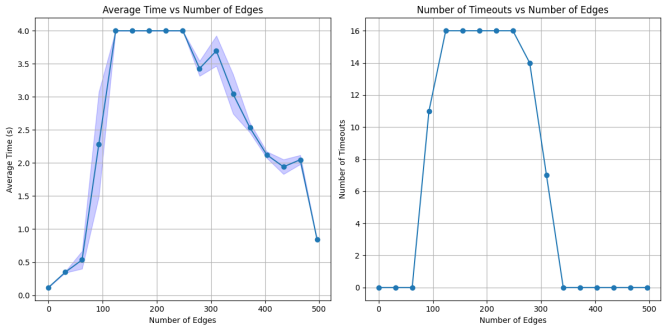

Figura 46. Gráfico de resultados (CP-SAT) para instâncias com 32 vértices e objetivo de 2 componentes conexos.

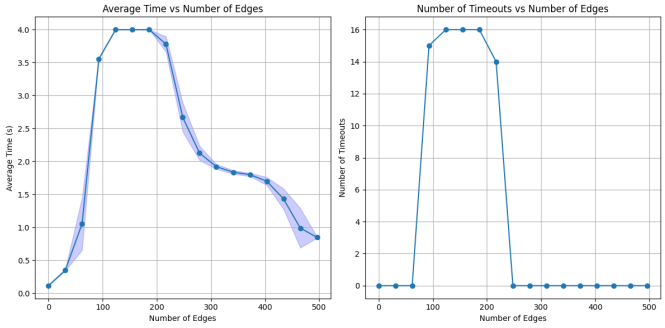

Figura 47. Gráfico de resultados (CP-SAT) para instâncias com 32 vértices e objetivo de 3 componentes conexos.

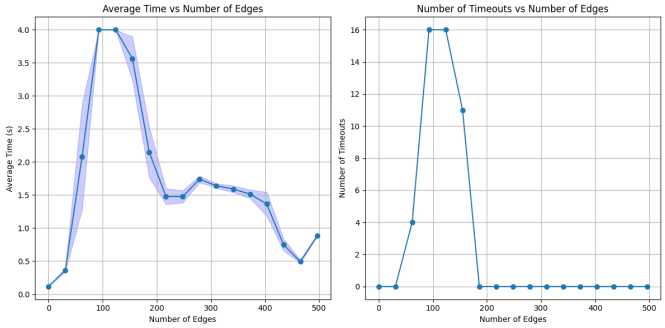

Figura 48. Gráfico de resultados (CP-SAT) para instâncias com 32 vértices e objetivo de 4 componentes conexos.

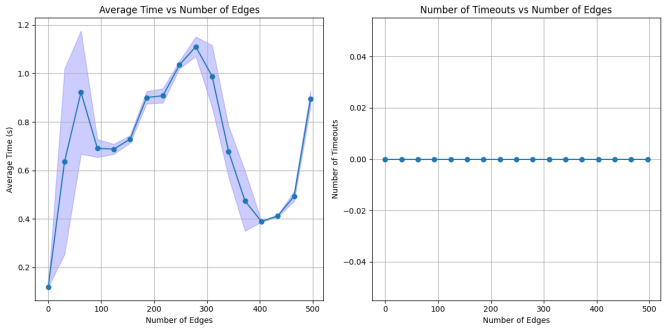

Figura 49. Gráfico de resultados (CP-SAT) para instâncias com 32 vértices e objetivo de 8 componentes conexos.

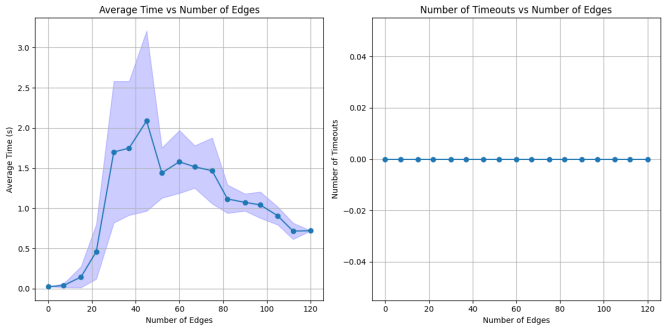

Figura 50. Resultados (PLI com M=n) para 16 vértices e 2 componentes.

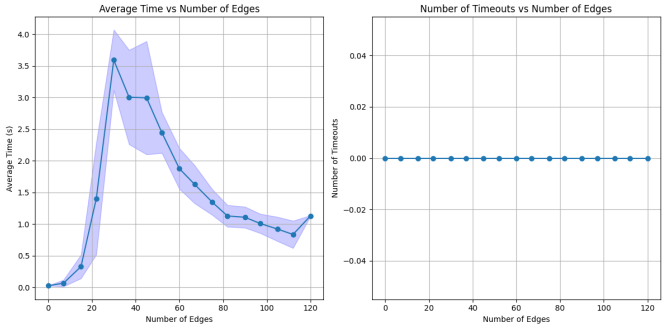

Figura 51. Resultados (PLI com M=n) para 16 vértices e 3 componentes.

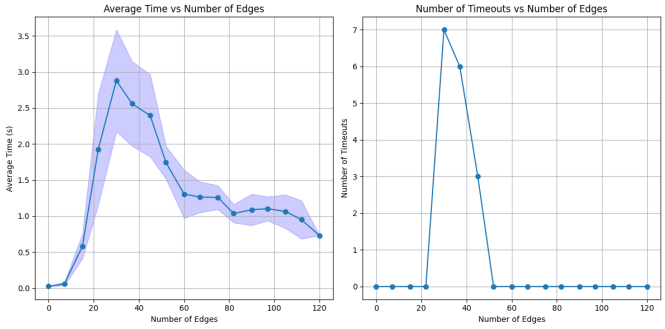

Figura 52. Resultados (PLI com M=n) para 16 vértices e 4 componentes.

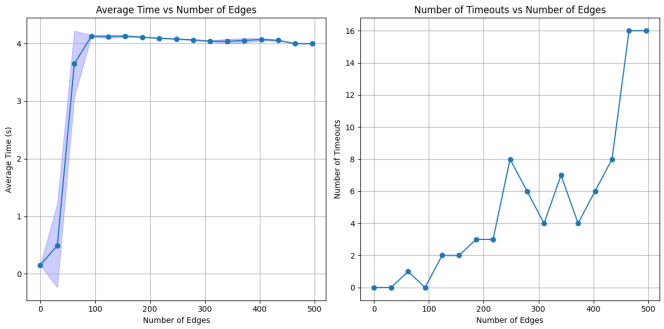

Figura 53. Resultados (PLI com M=n) para 32 vértices e 2 componentes.

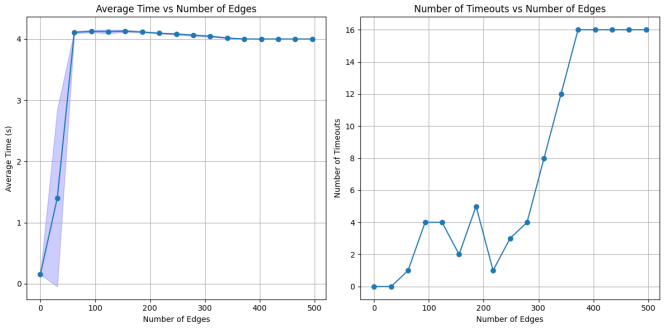

Figura 54. Resultados (PLI com M=n) para 32 vértices e 3 componentes.

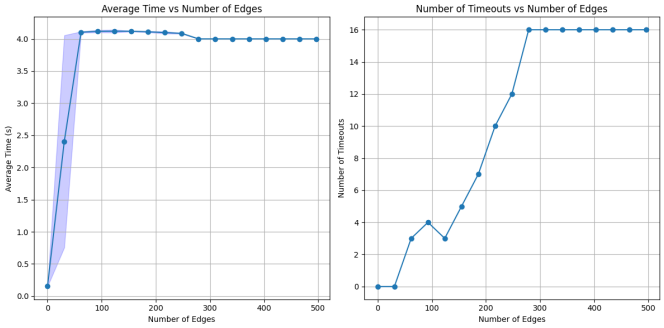

Figura 55. Resultados (PLI com M=n) para 32 vértices e 4 componentes.

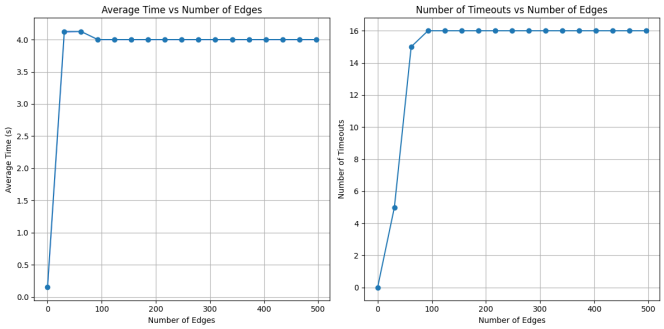

Figura 56. Resultados (PLI com M=n) para 32 vértices e 8 componentes.

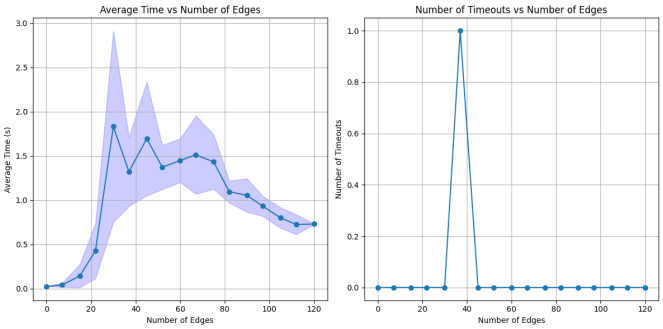

Figura 57. Resultados (PLI com k=c) para 16 vértices e 2 componentes.

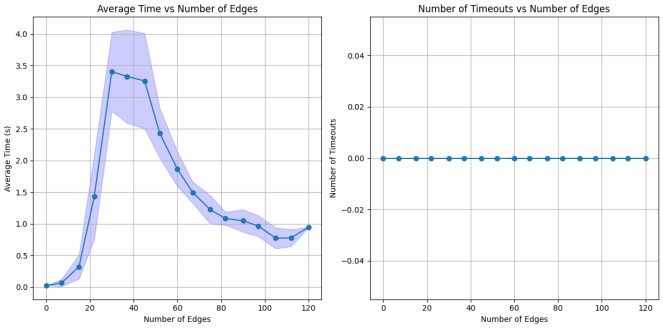

Figura 58. Resultados (PLI com k=c) para 16 vértices e 3 componentes.

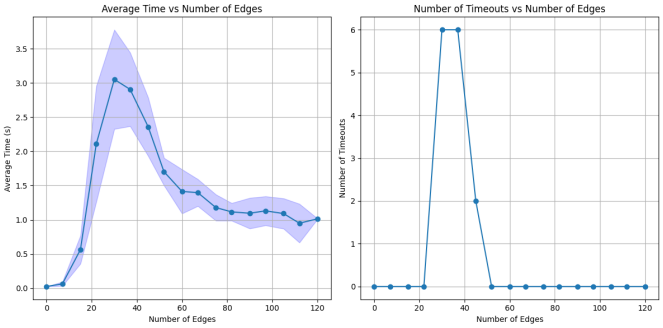

Figura 59. Resultados (PLI com k=c) para 16 vértices e 4 componentes.

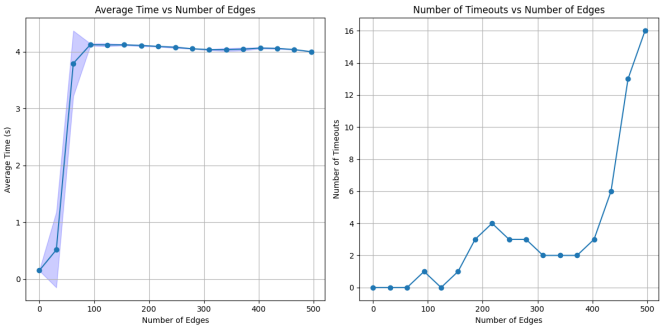

Figura 60. Resultados (PLI com k=c) para 32 vértices e 2 componentes.

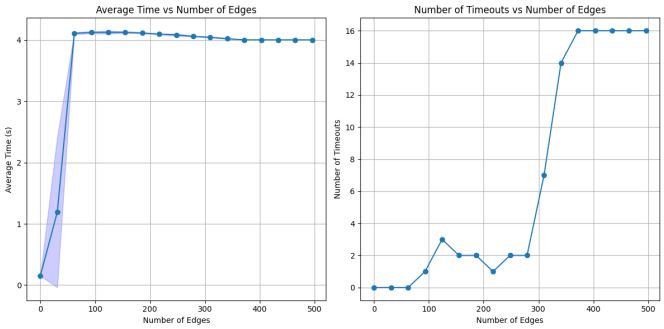

Figura 61. Resultados (PLI com k=c) para 32 vértices e 3 componentes.

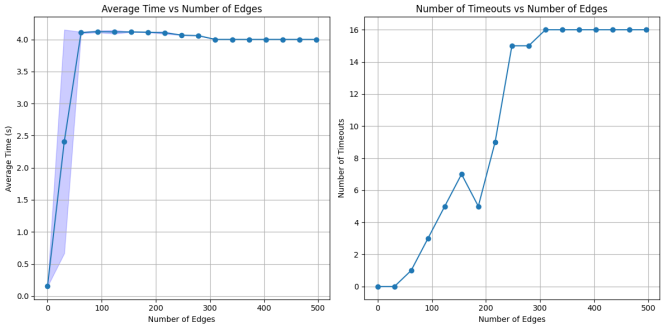

Figura 62. Resultados (PLI com k=c) para 32 vértices e 4 componentes.

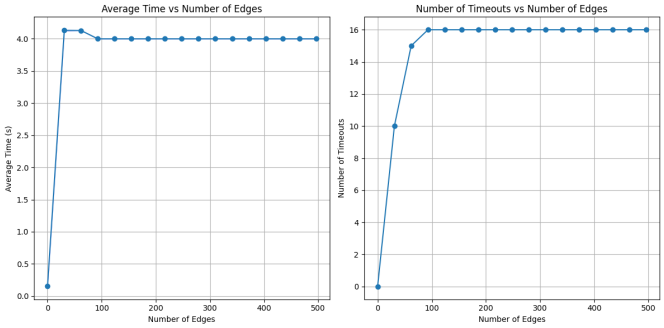

Figura 63. Resultados (PLI com k=c) para 32 vértices e 8 componentes.

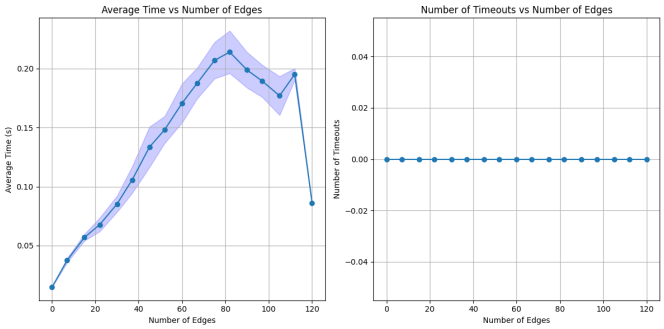

Figura 64. Resultados (CP-SAT com k=c) para 16 vértices e 2 componentes.

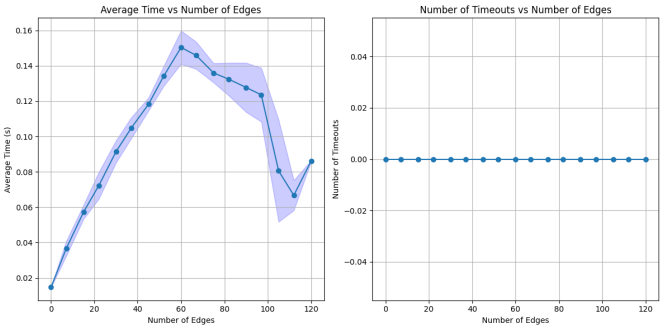

Figura 65. Resultados (CP-SAT com k=c) para 16 vértices e 3 componentes.

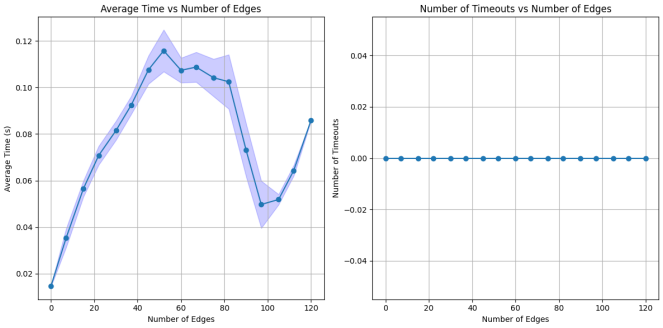

Figura 66. Resultados (CP-SAT com k=c) para 16 vértices e 4 componentes.

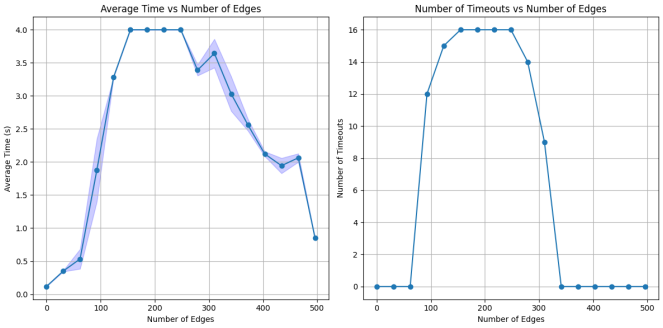

Figura 67. Resultados (CP-SAT com k=c) para 32 vértices e 2 componentes.

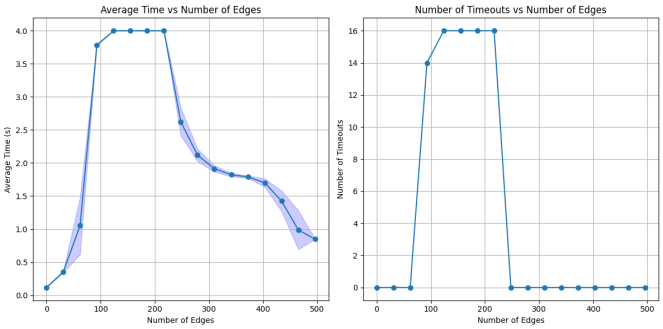

Figura 68. Resultados (CP-SAT com k=c) para 32 vértices e 3 componentes.

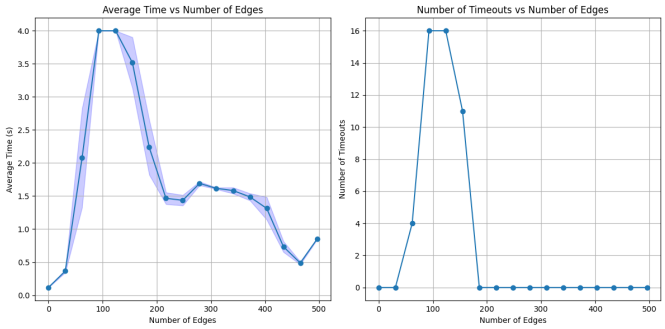

Figura 69. Resultados (CP-SAT com k=c) para 32 vértices e 4 componentes.

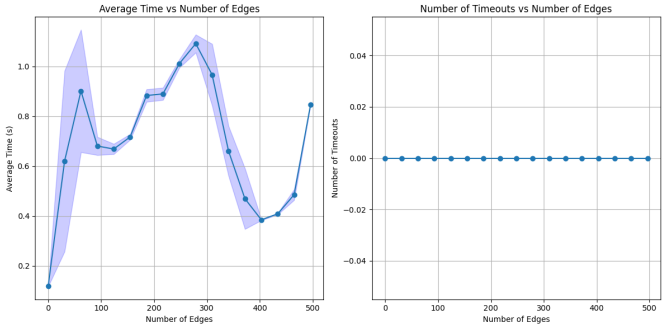

Figura 70. Resultados (CP-SAT com k=c) para 32 vértices e 8 componentes.

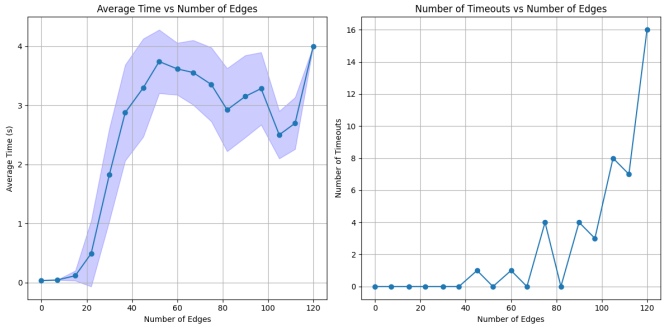

Figura 71. Resultados (PLI com quebra de simetria) para 16 vértices e 2 componentes.

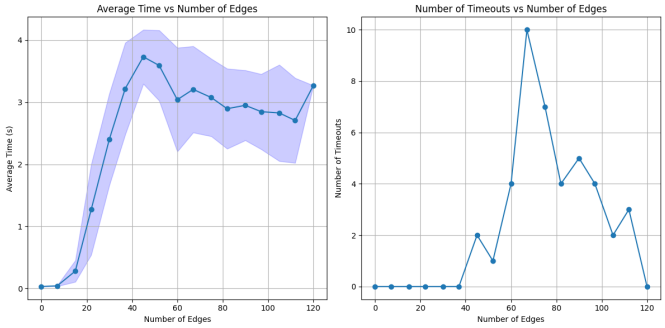

Figura 72. Resultados (PLI com quebra de simetria) para 16 vértices e 3 componentes.

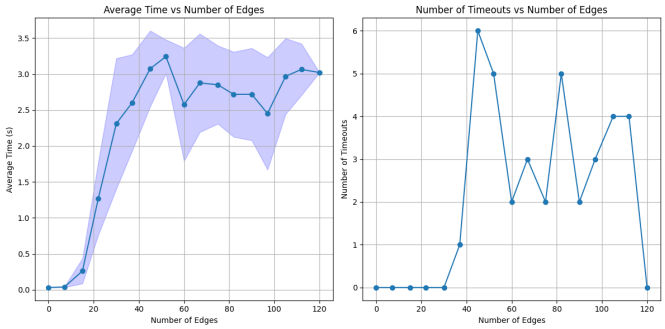

Figura 73. Resultados (PLI com quebra de simetria) para 16 vértices e 4 componentes.

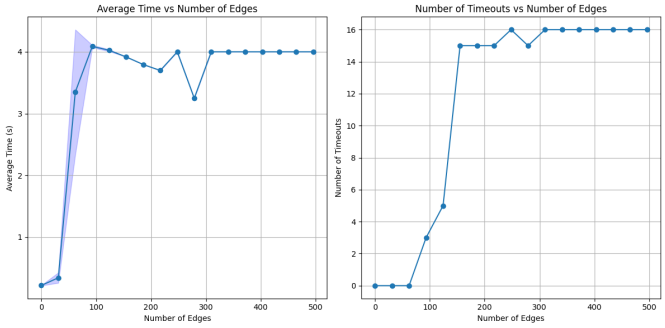

Figura 74. Resultados (PLI com quebra de simetria) para 32 vértices e 2 componentes.

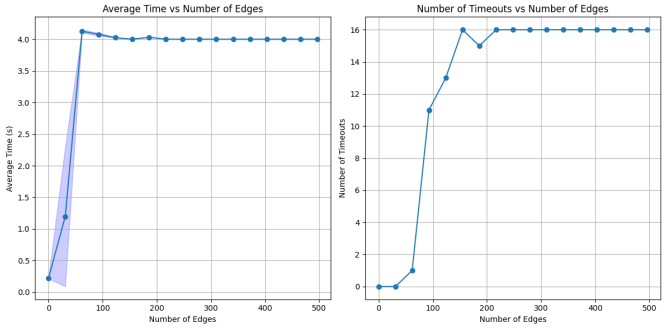

Figura 75. Resultados (PLI com quebra de simetria) para 32 vértices e 3 componentes.

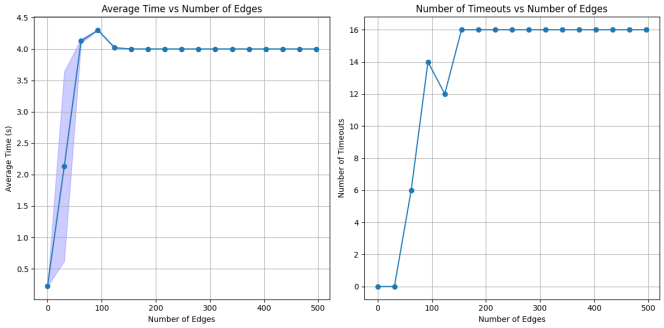

Figura 76. Resultados (PLI com quebra de simetria) para 32 vértices e 4 componentes.

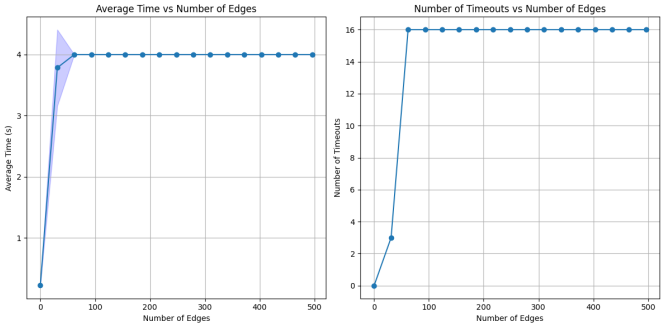

Figura 77. Resultados (PLI com quebra de simetria) para 32 vértices e 8 componentes.

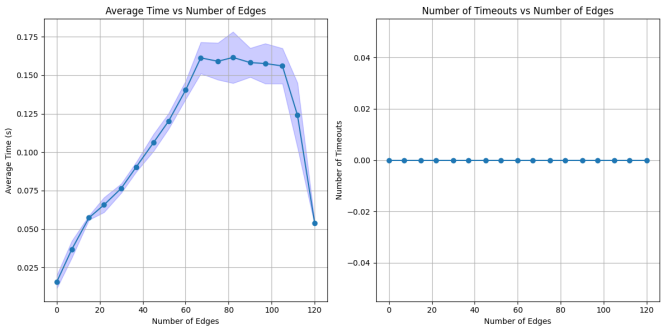

Figura 78. Resultados (CP-SAT com quebra de simetria) para 16 vértices e 2 componentes.

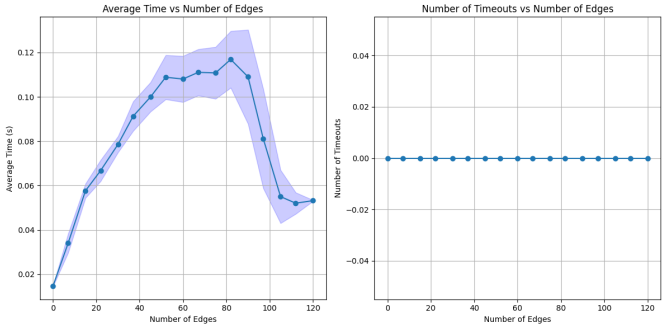

Figura 79. Resultados (CP-SAT com quebra de simetria) para 16 vértices e 3 componentes.

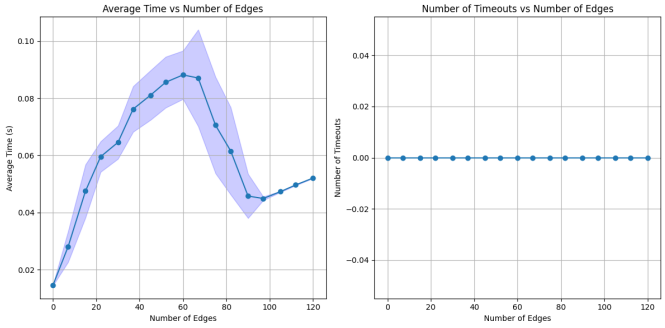

Figura 80. Resultados (CP-SAT com quebra de simetria) para 16 vértices e 4 componentes.

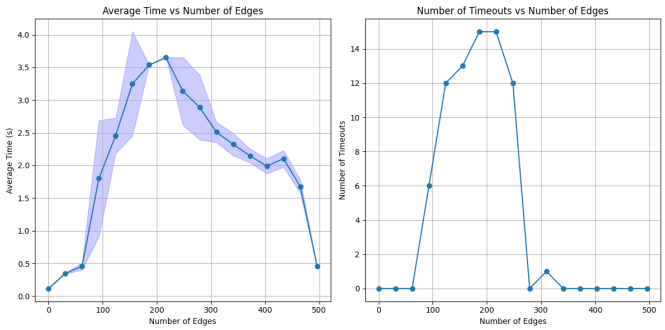

Figura 81. Resultados (CP-SAT com quebra de simetria) para 32 vértices e 2 componentes.

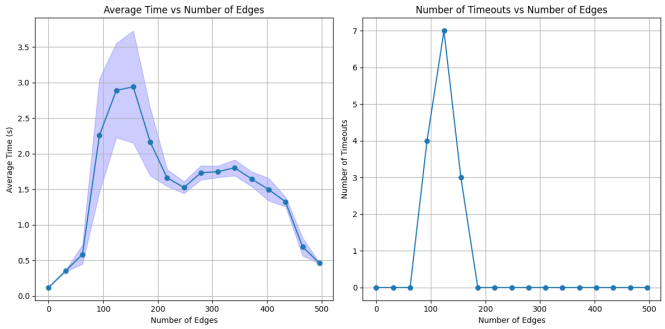

Figura 82. Resultados (CP-SAT com quebra de simetria) para 32 vértices e 3 componentes.

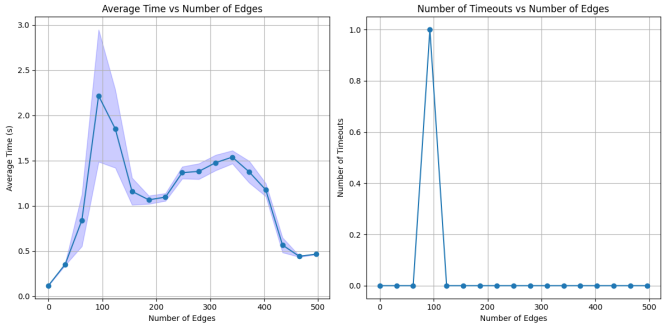

Figura 83. Resultados (CP-SAT com quebra de simetria) para 32 vértices e 4 componentes.

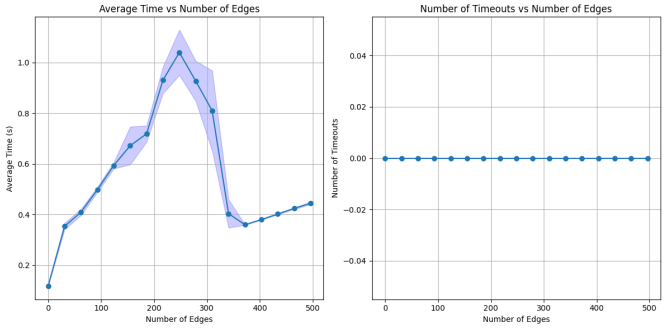

Figura 84. Resultados (CP-SAT com quebra de simetria) para 32 vértices e 8 componentes.

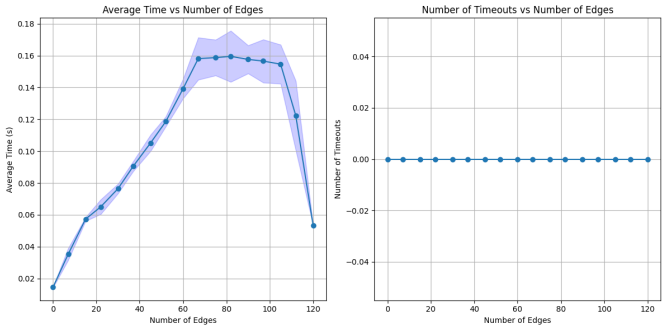

Figura 85. Resultados (CP-SAT com quebra de simetria simplificada) para 16 vértices e 2 componentes.

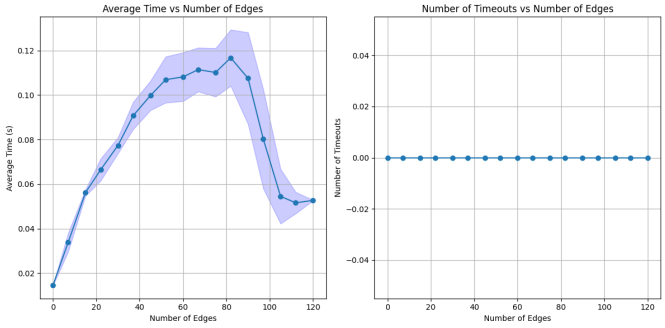

Figura 86. Resultados (CP-SAT com quebra de simetria simplificada) para 16 vértices e 3 componentes.

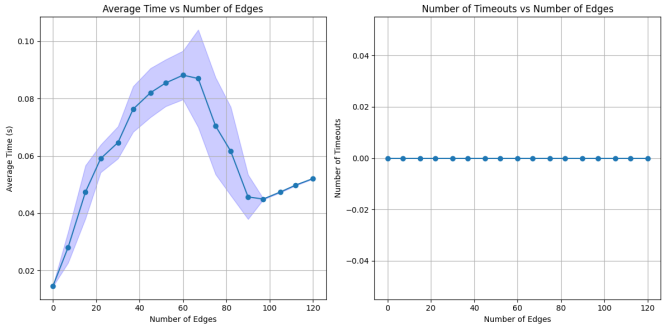

Figura 87. Resultados (CP-SAT com quebra de simetria simplificada) para 16 vértices e 4 componentes.

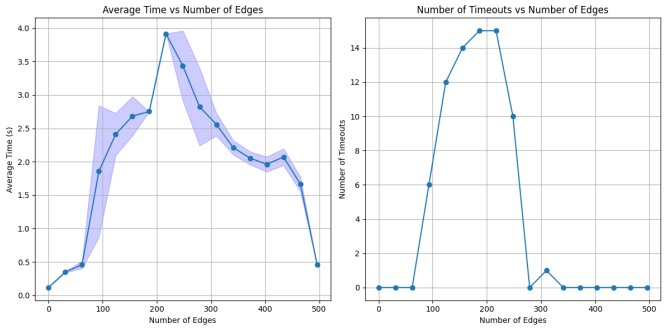

Figura 88. Resultados (CP-SAT com quebra de simetria simplificada) para 32 vértices e 2 componentes.

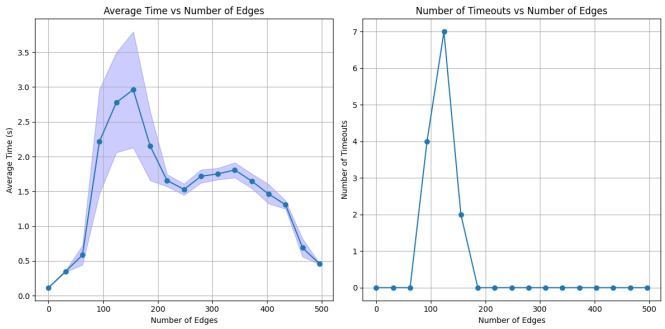

Figura 89. Resultados (CP-SAT com quebra de simetria simplificada) para 32 vértices e 3 componentes.

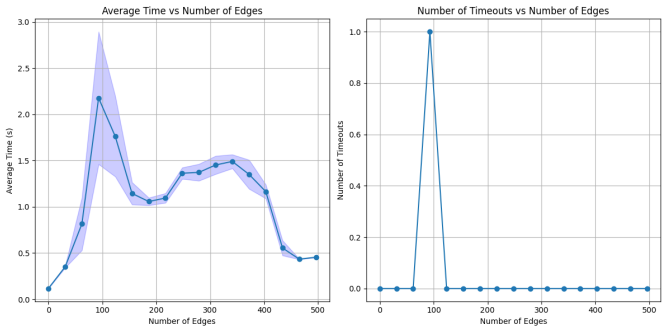

Figura 90. Resultados (CP-SAT com quebra de simetria simplificada) para 32 vértices e 4 componentes.

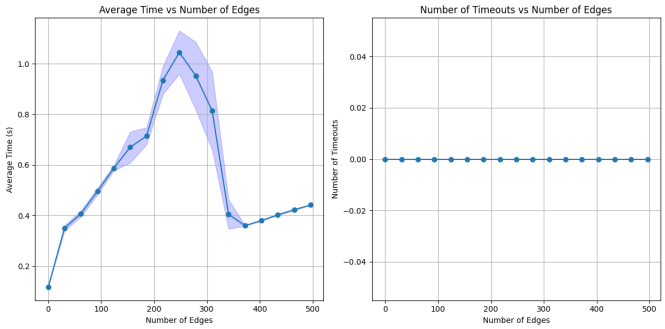

Figura 91. Resultados (CP-SAT com quebra de simetria simplificada) para 32 vértices e 8 componentes.