Comparative study of dual box covering algorithms for fractal graphs dimension computation

THÉO CANTALOUBE, ENSIMAG Student, Second year, France
CHING-LUEH CHANG, PI, Algorithms Laboratory, Yuan Ze University, Taiwan

This document covers all the research work done by Cantaloube Théo during his internship in Yuan Ze University about Dual box covering algorithms for fractal graphs dimension computation. First, we verify what has been done in box covering algorithm research. Second, we study two box covering problems (one of minimization, the other of maximization) and demonstrate different order relationships between their optimum. Third, we prove their $\mathcal{NP}$ complexity by reduction. Fourth, we prove their duality by their formulation into linear programs. Fifth, we consider few variants of the box covering problems. Sixth, we find greedy algorithms for these problems and prove none of these are approximations. Seventh, we formalize our fractal graphs. Eighth, we construct a framing procedure which allows us to exhibit an order relationship between the homothety and the box-counting dimensions. And finally, we apply these algorithms on fractal examples to verify their reliability.

CCS Concepts: • **Mathematics of computing** → *Approximation algorithms*; **Graph algorithms**; **Paths and connectivity problems**; • **Theory of computation** → *Computational complexity and cryptography*.

Additional Key Words and Phrases: Fractal dimension, Box counting, Complete box cover, Disjoint box cover, Duality

ACM Reference Format:
Théo Cantaloube and Ching-Lueh Chang. 2024. Comparative study of dual box covering algorithms for fractal graphs dimension computation. 1, 1 (August 2024), 33 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Contents

Authors' Contact Information: Théo Cantaloube, ENSIMAG Student, Second year, Grenoble, France; Ching-Lueh Chang, PI, Algorithms Laboratory, Yuan Ze University, Taoyuan, Taiwan.

ACM XXXX-XXXX/2024/8-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

1.1 Preamble

*"**Fractals** should be regarded in the same way as a biologist regards the definition of **Life**. There is no hard and fast definition, but just a list of properties characteristic of a living thing, such as the ability to reproduce or to move [...] In the same way, it seems best to regard a fractal as a set that has properties [...] rather than to look for a precise definition which will almost certainly exclude some interesting cases."* Falconer, [6, p. xxv]

Fractal objects indeed have many different definitions.

Self-similarity. The more commonly known definition is based on self-similarity [13]. Fractal objects are supposed to be composed

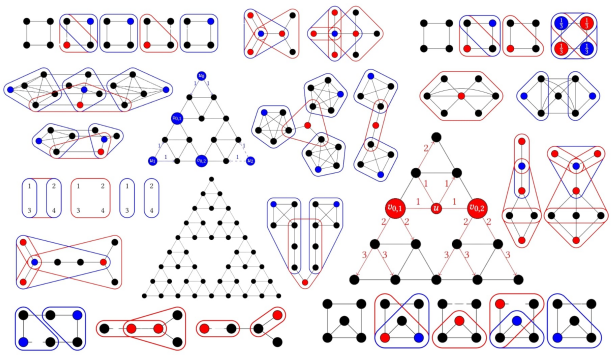

Fig. 1. In this paper, we will review or execute algorithms on all kinds of graphs. Many graph illustrations use Latex package Tikz Hypergraphs [19] created by Manuel Sorge in order to encircle vertices. More generally, in this paper's examples, when not precised, represented graphs are unweighted but be aware that all demonstrations are valid for weighted graphs. Indeed, weights only impact distance computation realized thanks to Dijkstra algorithm.

of samples of themselves. This self-similarity can be exact (like in the Sierpinski Triangle cf the left figure in Fig.2) or statistic (like in Trabecular bone structure cf the right figure in Fig.2). Self-similarity can be observed in things as simple as ferns or roumanesco broccoli...

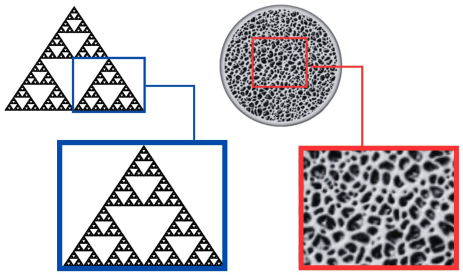

Fig. 2. Comparison between exact and statistic self-similarity (Picture: www.morelandobgyn.com/blog/)

Scale-free complexity. Another definition for fractal objects is based on scale-free complexity. Fractal objects are supposed to exhibit detailed structure at every scale. Here again, scale-free complexity can be exact (like in the Mandelbrot fractal cf the left figure in Fig.3) or statistic (like in the Internet network graph cf the right figure in Fig.3). Scale-free complexity can be observed in things as simple as the coast of Great Britain.

Fractal dimension. Finally, the last definition (which interests us) is based on fractal dimension. Fractal dimension has the peculiarity of not always being integral. It has no universal definition, but rather dozens of definitions: Information dimension [22], Correlation dimension [5], Generalized or Rényi dimensions [3], Higuchi dimension, Lyapunov dimension, Packing dimension, Assouad dimension... In our study context, not all of them are relevant. Many

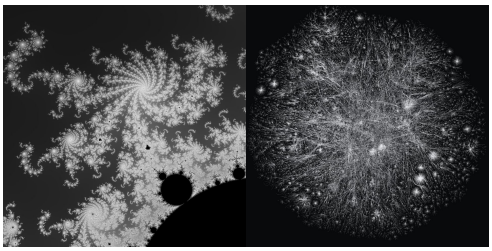

Fig. 3. Comparison between exact and statistic scale-free complexity (Pictures: https://www.flickr.com/photos/dominicspics/5420891562, www.opte.org/maps/)

are used in more physics-related frameworks, such as chaos theory or information theory, for example. In this paper, we will focus on 2 different fractal dimensions: Hausdorff dimension (equivalent to Homothety dimension) and Box-counting dimension.

Hausdorff dimension. According to Mandelbrot's definition (a Franco-American mathematician known for his pioneering work in the field of fractals, a term he coined), a fractal is a set for which the Hausdorff D_H dimension strictly exceeds the topological dimension D_T [Mandelbrot 1983 [14]]. Topological dimension D_T corresponds to the dimension in which an object can intuitively be described (example: topological dimension of a line is 1, topological dimension of a surface is 2...). Let's defined the quantity:

$$H_r^s(X) = \inf_{\text{diam}(A_i)<r} \left\{ \sum_{i=1}^{+\infty} \text{diam}(A_i)^s \mid X \subseteq \bigcup_{i=1}^{+\infty} A_i \right\}$$

And then the Hausdorff measure $H^s(X)$ of the set X as follows:

$$H^s(X) = \lim_{r \to 0^+} H_r^s(X)$$

Finally, Hausdorff dimension is defined as follows:

$$D_H(X) = \inf \left\{ s, H^s(X) = 0 \right\} = \sup \left\{ s, H^s(X) = \infty \right\}$$

It's the most rigorous definition, valid for any set but very complicated to implement. An equivalent definition exists, such as the homothety dimension D_h. This is the simplest translation of Hausdorff's dimension, applicable to fractal sets with internal disjoint homotheties. D_h is defined as the solution of the following equation:

$$\sum_{k=1}^{N} r_k^{D_h} = 1$$

where N is the number of homotheties and r_k the homothety ratios. In case of sets with internal homotheties of the same ratio r, we obtain the homothety dimension D'_h:

$$N r^{D_{h'}} = 1 \Leftrightarrow D_{h'} = \frac{ln(N)}{ln(\frac{1}{r})}$$

A simple application is described in Fig.4. On the upper side, we have three simple geometrical objects (a line L, a surface S and a volume V). By multiplying all their dimensions by 2 (the inverse of the homothety ratio), we obtain 2, 4 and 8 new objects of the original size (the number of homotheties). On the lower side, we have a portion of the Koch snowflake fractal F. By multiplying all their dimensions by 3 (the inverse of the homothety ratio), we obtain

4 new objects of the original size (the number of homotheties). We collect all this informations and compute homothety dimension in Tab. 1.

Object X	Line	Surface	Volume	Koch curve
Homotheties number N	2	4	8	4
Homothety ratio r	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
$D_T(X)$	1	2	3	1
$D'_h(X) = \ln_{\frac{1}{r}}(N)$	1	2	3	1.226...

Table 1. Homothety dimension computation

This example confirms the Mandelbrot definition. Hausdorff dimension coincides with topological dimension if sets are not fractal. In this paper, we'll indifferently use the notations D_H, D_h and D'_h (since they are equivalent).

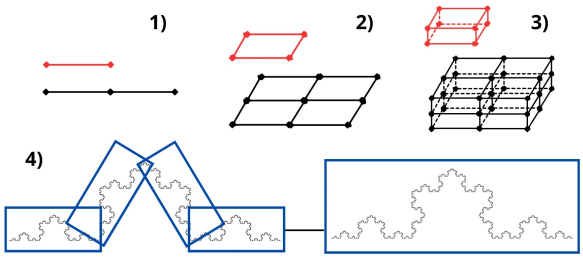

Fig. 4. Homothety dimension of geometrical objects and fractal

Box Counting dimension. The last fractal dimension definition we will study is named the Box Counting dimension D_{box}. Let's denote $N(\varepsilon)$ the required number of subsets of diameter ε to cover the set. We adopt the following heuristic: $N(\varepsilon)$ is "approximately equivalent" to $c\varepsilon^{-D_{\text{box}}}$ with c a positive constant. By composing this equivalence by ln, we obtain $\ln(N(\varepsilon)) = \ln(c) - D_{\text{box}} \ln(\varepsilon)$. Finally, we can express D_{box} as follows:

$$D_{\text{box}} = \lim_{\varepsilon \to 0^+} \frac{\ln(N(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

Simple computations of $N(\varepsilon)$ are described in Fig.5.

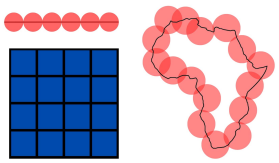

Fig. 5. Box counting method for geometrical objects and the Africa coast

If D_{box} limit doesn't exist, we usually denote:

$$D_{\overline{\text{box}}} = \overline{\lim_{\varepsilon \to 0^+}} \frac{\ln(N(\varepsilon))}{\ln(\frac{1}{\varepsilon})} \quad \text{and} \quad D_{\underline{\text{box}}} = \underline{\lim_{\varepsilon \to 0^+}} \frac{\ln(N(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

Box-counting dimension computation remains the same for all these different types of covers [6]:

1. number of squares in a ε-grid covering the set
2. smallest number of balls of radius ε covering the set
3. smallest number of cubes of side ε covering the set
4. smallest number of subsets of diameter ε covering the set
5. largest number of disjoint ε-balls centered on the set

In this paper will be studied the duality between box-counting 2 and box-counting 5. Finally, the following inequality as been proven [6]: $D_T \leq D_H \leq D_{\text{box}}$. In case of exact self-similarity, D_H and D_{box} are conjectured to be equal [16].

1.2 State of art

Aside of geometrical objects, fractal dimension has also been studied in graphs. (Be aware to not confuse fractal dimension for graphs with dimension for graphs (the least integer n such that the graph with edges having unit length can be represented in n dimensions)). Equivalents of Hausdorff dimension have been found [17] but it still remains not easy to manipulate.

On the other hand, box-counting algorithms for graphs were searched for a long time [9], [10], [21]. Instead of balls of radius ε or subsets of diameter ε, we search subsets of vertices such that the distance from each vertex of the box to its center is at least ε or subsets of vertices such that the distance between each vertex of the box is at least ε.

Lot of different algorithms already have been developed. Song et al. experimented four algorithms: the greedy coloring (GC) algorithm, the random-burning (RB) algorithm, the compact-box-burning (CBB) algorithm and the maximum-exclude-mass-burning (MEMB) algorithm [18]. First comes from a direction association found between box covering instances and vertex coloring instances (which could have been used to prove $\mathcal{NP}$-complexity of box covering problems in section 3). Second is a random greedy method that failed to reach box covers values close to this minimum (and which implies the need to base algorithms on heuristics that make more sense than randomness). Third idea is to generate boxes by randomly selecting node in boxes neighborhood until the cover is complete (which focus on ε-diameter boxes instead of ε-radius boxes). Finally, fourth strategy tries to locate some optimal box centers nodes (that's quite the idea of algorithms developed in section 6). At the exception of the random method, these algorithms show similar results.

Progress has been made with new heuristics based on burning methods that outperformed previous algorithms [15], [20]. There were also the Ratio of excluded mass to closeness centrality algorithm [24], the Merge algorithm [12], the Simulated annealing algorithm [12], the Differential evolution algorithm [11] or the Fuzzy box algorithm [23] but no polynomial method exists to date.

It has to be noted that box-covering algorithms have other applications than fractal dimension computation. For example, it's useful

to determinate how to optimally cover networks (for telephony, electricity, internet...).

The novelty of this paper compared to the current state of the art consists in the study of the disjoint box cover problem and its relation with the usual complete box cover problem. Instead of basing our work solely on an empirical approach, we will formalize the internal characteristics of box-counting algorithms in an academic way, in order to discover interesting properties.

2 Problems

First, let's define for a given graph $G = (V, E)$, its distance function $dist : V \times V \to \mathbb{R}$ computed thanks to Dijkstra algorithm, a given $\varepsilon \in \mathbb{R}$ and a given vertex $u \in V$, the box $\mathcal{B}(u, \varepsilon)$:

$$\mathcal{B}(u, \varepsilon) = \{v \in V \mid dist(u, v) \leq \varepsilon\}$$

Thanks to *dist* symmetry, we have this fundamental first property:

Proposition 2.1. $\forall\, u, v \in V,\ u \in \mathcal{B}(v, \varepsilon) \Leftrightarrow v \in \mathcal{B}(u, \varepsilon)$

Proof. $u \in \mathcal{B}(v, \varepsilon) \Leftrightarrow \text{dist}(v, u) \leq \varepsilon$
$\Leftrightarrow \text{dist}(u, v) \leq \varepsilon \Leftrightarrow v \in \mathcal{B}(u, \varepsilon)$ □

We introduce two fundamental sets $\mathcal{D}(G, \varepsilon)$ and $\mathcal{C}(G, \varepsilon)$:

$$\mathcal{D}(G, \varepsilon) = \{D \subseteq V \mid (\mathcal{B}(d, \varepsilon))_{d \in D} \text{ are pairwise disjoint. }\}$$

$$\mathcal{C}(G, \varepsilon) = \{C \subseteq V \mid V = \textstyle\bigcup_{c \in C} \mathcal{B}(c, \varepsilon)\}$$

These sets are all box covers fulfilling the condition of disjunction / coverage. All new notations of this paper are referenced in the section 11.

2.1 Box Cover problems

Complete Box Cover decision problem:

- Input: $G = (V, E)$, $\varepsilon \in \mathbb{R}$, $k \in \mathbb{N}$
- Output: Is there a vertex set $C \in \mathcal{C}(G, \varepsilon)$ such that $|C| = k$?

Complete Box Cover minimization problem:

- Input: $G = (V, E)$, $\varepsilon \in \mathbb{R}$
- Output: What is the size of the smaller set $C \in \mathcal{C}(G, \varepsilon)$?

Disjoint Box Cover decision problem:

- Input: $G = (V, E)$, $\varepsilon \in \mathbb{R}$, $k \in \mathbb{N}$
- Output: Is there a vertex set $D \in \mathcal{D}(G, \varepsilon)$ such that $|D| = k$?

Disjoint Box Cover maximization problem:

- Input: $G = (V, E)$, $\varepsilon \in \mathbb{R}$
- Output: What is the size of the greater set $D \in \mathcal{D}(G, \varepsilon)$?

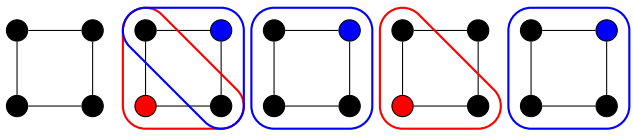

Fig. 6. a given graph G and box covers corresponding to $\text{CBC}(G, 1)$, $\text{CBC}(G, 2)$, $\text{DBC}(G, 1)$ and $\text{DBC}(G, 2)$

Some obvious properties. These problems are also called in this paper CBC-DEC, CBC, DBC-DEC and DBC. Some simple examples are provided in Fig.6. We should notice that:

Proposition 2.2. $\textit{CBC-DEC}(\cdot, \cdot, |V|) = \textit{DBC-DEC}(\cdot, \cdot, 1) = \textit{true}$

Proof. $V \in \mathcal{C}(G, \varepsilon)$ and $\forall\, u \in V,\ \{u\} \in \mathcal{D}(G, \varepsilon)$. □

That means that CBC and DBC have always an answer and that gives us bounds for box centers. It is very useful for proving the existence of feasible solutions to Linear Programs derived from these minimization / maximization problems, cf Section 4.1. We also have $\forall\, k \in [[1, |V|]]$:

Proposition 2.3.

$$\textit{CBC-DEC}(G, \varepsilon, k) = \textit{true} \Rightarrow \textit{CBC-DEC}(G, \varepsilon, k+1) = \textit{true}$$

Proposition 2.4.

$$\textit{DBC-DEC}(G, \varepsilon, k) = \textit{true} \Rightarrow \textit{DBC-DEC}(G, \varepsilon, k-1) = \textit{true}$$

Proposition 2.5. $\textit{CBC-DEC}(G, \varepsilon, k) = \textit{true} \Leftrightarrow \textit{CBC}(G, \varepsilon) \leq k$

Proposition 2.6. $\textit{DBC-DEC}(G, \varepsilon, k) = \textit{true} \Leftrightarrow \textit{DBC}(G, \varepsilon) \geq k$

We introduce two more sets $\overline{\mathcal{D}}(G, \varepsilon)$ and $\overline{\mathcal{C}}(G, \varepsilon)$ which will be useful later:

$$\overline{\mathcal{D}}(G, \varepsilon) = \{D \in \mathcal{D}(G, \varepsilon) \mid \text{DBC}(G, \varepsilon) = |D|\}$$

$$\overline{\mathcal{C}}(G, \varepsilon) = \{C \in \mathcal{C}(G, \varepsilon) \mid \text{CBC}(G, \varepsilon) = |C|\}$$

We can observe that, whatever the inputs, we have always DBC ≤ CBC. Let's prove it.

2.2 Comparison between DBC and CBC

Theorem 2.1. $\forall\, (D, C) \in \mathcal{D}(G, \varepsilon) \times \mathcal{C}(G, \varepsilon),\ |D| \leq |C|$

Proof. Let's take a given $\varepsilon \in \mathbb{R}$ and a given $G = (V, E)$.
For every $(D, C) \in \mathcal{D}(G, \varepsilon) \times \mathcal{C}(G, \varepsilon)$, we create a "matching function" $f_{D,C} : D \to \mathcal{P}(C)$ defined as follows:

$$\forall\, d \in D,\ f_{D,C}(d) = \{c \in C \mid d \in \mathcal{B}(c, \varepsilon)\}$$

Let's prove the following lemmas:

Lemma 2.2. $\forall\, d \in D,\ \left|f_{D,C}(d)\right| \geq 1$

Proof. $\forall\, d \in D,\ d \in V = \bigcup_{c \in C} \mathcal{B}(c, \varepsilon) \Rightarrow$
$\forall\, d \in D,\ \exists\, c \in C \mid d \in \mathcal{B}(c, \varepsilon) \Rightarrow \forall\, d \in D, |f_{D,C}(d)| \geq 1$ □

Lemma 2.3. $\forall\, d \neq d' \in D,\ f_{D,C}(d) \cap f_{D,C}(d') = \emptyset$

Proof. Let's suppose that:

$$\exists\, d, d' \in D \mid f_{D,C}(d) \cap f_{D,C}(d') \neq \emptyset \Rightarrow$$

$$\exists\, c \in C \mid d, d' \in \mathcal{B}(c, \varepsilon) \underset{\text{Prop. 2.1}}{\Rightarrow} \exists\, c \in C \mid c \in \mathcal{B}(d, \varepsilon) \cap \mathcal{B}(d', \varepsilon)$$

$$\Rightarrow \mathcal{B}(d, \varepsilon) \cap \mathcal{B}(d', \varepsilon) \neq \emptyset \Rightarrow d = d' (D \text{ is a disjoint cover})$$

It's a bit like an intersection injectivity property. □

Finally, we can obtain the equation:

$$|D| = \sum_{d \in D} 1 \underset{\text{Lem. 2.2}}{\leq} \sum_{d \in D} \left|f_{D,C}(d)\right| \underset{\text{Lem. 2.3}}{=} \left|\biguplus_{d \in D} f_{D,C}(d)\right|$$

$$= \left|\biguplus_{c \in f_{D,C}(D)} c\right| = \left|f_{D,C}(D)\right| \leq |C|$$ □

Corollary 2.3.1. $DBC(\cdot,\cdot) \leq CBC(\cdot,\cdot)$

Proof. Th. 2.1 is equivalent to: $\max_{D\in\mathcal{D}(G,\varepsilon)} |D| \leq \min_{C\in\mathcal{C}(G,\varepsilon)} |C|$
Or, put another way:
$$\forall\, G=(V,E),\ \varepsilon\in\mathbb{R},\ \mathrm{DBC}(G,\varepsilon)\leq \mathrm{CBC}(G,\varepsilon)$$ □

And that leads us to consider the following problem: for which kind of graphs G do we have $\mathrm{DBC}(G,\varepsilon)=\mathrm{CBC}(G,\varepsilon)$?

2.3 Equality between DBC and CBC

$\overline{f_{D,C}}$ bijection. Now, we're going to characterize the equality between DBC and CBC.

Theorem 2.4. $\forall\, G=(V,E),\ \varepsilon\in\mathbb{R},\ (DBC(G,\varepsilon)=CBC(G,\varepsilon))$
$\Leftrightarrow \forall\,(D,C)\in\overline{\mathcal{D}}(G,\varepsilon)\times\overline{\mathcal{C}}(G,\varepsilon),\ f_{D,C}$ *is bijective.*

Proof. Let's take a given $\varepsilon\in\mathbb{R}$ and a given $G=(V,E)$.
$DBC(G,\varepsilon)=CDC(G,\varepsilon)\Rightarrow\forall\,(D,C)\in\overline{\mathcal{D}}(G,\varepsilon)\times\overline{\mathcal{C}}(G,\varepsilon)\mid |D|=|C|$. Therefore, final equation in proof of Th. 2.1 becomes:

$$|D|=\sum_{d\in D}1=\sum_{d\in D}|f_{D,C}(d)|=|C|\Rightarrow\forall\, d\in D,\ |f_{D,C}(d)|=1$$

So since every image set contains one vertex, we can extract from $f_{D,C}$ a matching function which returns for every vertex of D the unique image vertex of D: $\overline{f_{D,C}}: D\to C$. Lem. 2.3 becomes:

$$\exists\, d,d'\in D\mid \overline{f_{D,C}(d)}=\overline{f_{D,C}(d')}\Rightarrow$$

$$\exists\, d,d'\in D\mid f_{D,C}(d)\cap f_{D,C}(d')\neq\emptyset\Rightarrow d=d'$$

The intersection injectivity property for $f_{D,C}$ is a simple injectivity property for $\overline{f_{D,C}}$. Thanks to the Dirichlet's drawer principle, the injectivity property and the equality between the size of D and C implies that $\overline{f_{D,C}}$ is a bijection and $\overline{f_{C,D}}\left(=\overline{f_{D,C}}^{-1}\right)$ is well-defined. In simple terms, each center of a disjoint box corresponds exactly to one center of a complete box and vice versa. □

In the bottom of Fig. 7, we show a example of such a bijection between a disjoint box cover and a complete box cover.

Since we found a bijection between every complete box cover and every disjoint box cover, we could try to create a bijection between complete box covers tuples or disjoint box covers tuples. They would be defined by $f_{C,C'}=f_{D,C'}\circ f_{C,D}$ or $f_{D,D'}=f_{C,D'}\circ f_{D,C}$ given the fact that $f^{-1}_{C,C'}=(f_{D,C'}\circ f_{C,D})^{-1}=f^{-1}_{C,D}\circ f^{-1}_{D,C'}=f_{D,C}\circ f_{C',D}=f_{C',C}$ and $f^{-1}_{D,D'}=(f_{C,D'}\circ f_{D,C})^{-1}=f^{-1}_{D,C}\circ f^{-1}_{C,D'}=f_{C,D}\circ f_{D',C}=f_{D',D}$. Problem is that $f_{C,C'}$ and $f_{D,D'}$ notations aren't viable because they are not unique. By choosing a different D for $f_{C,C'}$ or a different C for $f_{D,D'}$, you would have a totally different bijection. But all the same, bijections exist between same category box covers tuples for which each center of a box corresponds exactly to one center of box in the other cover and vice versa.

DCBC-DEC decision problem. We can extract from the equality of DBC and CBC a decision problem called is this paper DCBC-DEC:

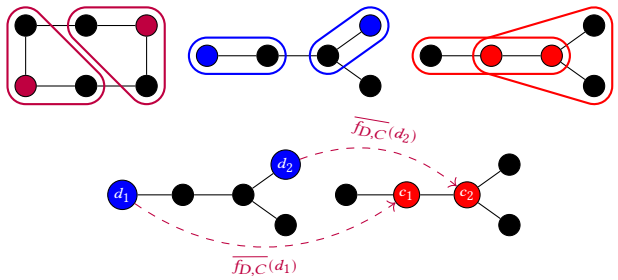

Fig. 7. On the upper left: a graph G with a box cover $\in\mathcal{D}(G,1)\cap\mathcal{C}(G,1)$. On the upper right: a graph G' with a disjoint box cover $D\in\mathcal{D}(G',1)$ and a complete box cover $C\in\mathcal{C}(G',1)$. At the bottom: the $\overline{f_{D,C}}$ bijection

Disjoint Complete Box Cover decision problem:

- Input: $G=(V,E)$, $\varepsilon\in\mathbb{R}$
- Output: Is there a vertex set $B\in\mathcal{D}(G,\varepsilon)\cap\mathcal{C}(G,\varepsilon)$?

The first box cover in Fig.7 is a solution of $\mathrm{DCBC}(G,1)$. Mathematically, we have this obvious implication:

Theorem 2.5. $DCBC(G,\varepsilon)=true\Rightarrow DBC(G,\varepsilon)=CBC(G,\varepsilon)$

Proof. $\mathrm{DCBC}(G,\varepsilon)=\text{true}\Leftrightarrow\exists\, B\in\mathcal{D}(G,\varepsilon)\cap\mathcal{C}(G,\varepsilon)\Rightarrow$

$$\mathrm{DBC}(G,\varepsilon,|B|)=\mathrm{CBC}(G,\varepsilon,|B|)=\text{true}\underset{\text{Prop. 2.6 and }Prop.2.5}{\Rightarrow}$$

$$\mathrm{DBC}(G,\varepsilon)\geq|B|\text{ and }\mathrm{CBC}(G,\varepsilon)\leq|B|\Rightarrow\mathrm{DBC}(G,\varepsilon)\geq\mathrm{CBC}(G,\varepsilon)$$

$$\mathrm{DBC}(G,\varepsilon)\underset{\text{Cor. 2.3.1}}{\leq}\mathrm{CBC}(G,\varepsilon)\Rightarrow\mathrm{DBC}(G,\varepsilon)=\mathrm{CBC}(G,\varepsilon)$$ □

Intuitively, since box covers both disjoint and complete implies the equality of *DBC* and *CBC* (Th. 2.5), we could think that the equality of *DBC* and *CBC* implies to the existence of a box cover both complete and disjoint (contrapositive of the reciprocal of Th. 2.5) so we get the equivalence. Unfortunately, that's false. In Fig.7, $\mathrm{DBC}(G',1)=\mathrm{CBC}(G',1)(=2)$ but it does not exist a box cover $\in\mathcal{D}(G',1)\cap\mathcal{C}(G',1)$.

In Fig.8, we illustrate this situation. We see that:

$$\mathcal{D}(G,\varepsilon)\cap\mathcal{C}(G,\varepsilon)=\overline{\mathcal{D}}(G,\varepsilon)\cap\overline{\mathcal{C}}(G,\varepsilon)$$

But we know that:

$$\mathrm{DBC}(G,\varepsilon)=\mathrm{CBC}(G,\varepsilon)\nRightarrow\mathcal{D}(G,\varepsilon)\cap\mathcal{C}(G,\varepsilon)\neq\emptyset$$

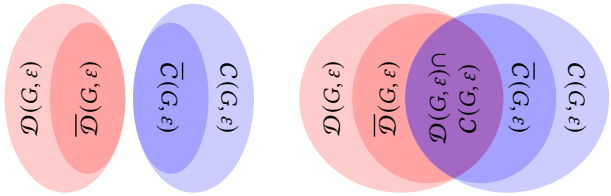

Fig. 8. On the left: box cover partition when disjoint and compact box covers intersection is empty. On the right: box cover partition when disjoint and compact box covers intersection isn't empty. If DCBC is true, we're on the right situation. If we're on the right situation, DBC and CBC are equal. But if we're on the left situation, DBC and CBC aren't necessary different.

DCBC characterization. Now, we're going to characterize graphs solution for the DCBC problem. Box centers b_1 and b_2 are called *ε-consistent* if:

$$\forall\ (u_1, u_2) \in \mathcal{B}(b_1, \varepsilon) \times \mathcal{B}(b_2, \varepsilon) \cap E,$$

$$w(u_1, u_2) + \max(\text{dist}(u_1, b_1),\ \text{dist}(u_2, b_2)) > \varepsilon$$

Where $w(u_1, u_2)$ is the weight of the edge (u_1, u_2) (1 if the graph is unweighted). A box cover $B \subseteq V$ is *ε-consistent* if:

$$\forall\ b_1 \neq b_2 \in B,\ b_1 \text{ and } b_2 \text{ are } \varepsilon\text{-consistent}$$

Finally, we call $\mathcal{D}'(G, \varepsilon)$ the set of ε-consistent box covers.

Theorem 2.6. $\mathcal{D}(G, \varepsilon) \subset \mathcal{D}'(G, \varepsilon)$

Proof. $D \notin \mathcal{D}'(G, \varepsilon) \Leftrightarrow D$ is ε-inconsistent.

$$\Leftrightarrow \exists\ d_1, d_2 \in D \mid\ d_1 \text{ and } d_2 \text{ are } \varepsilon\text{-inconsistent}$$

$$\Leftrightarrow \exists\ (u_1, u_2) \in \mathcal{B}(d_1, \varepsilon) \times \mathcal{B}(d_2, \varepsilon) \cap E \mid$$

$$w(u_1, u_2) + \max(\text{dist}(u_1, d_1),\ \text{dist}(u_2, d_2)) \leq \varepsilon \Leftrightarrow$$

$$w(u_1, u_2) + \text{dist}(u_1, d_1) \leq \varepsilon \text{ or } w(u_1, u_2) + \text{dist}(u_2, d_2) \leq \varepsilon$$

Without loss of generality, let's say we're in the first scenario. Thanks to the triangular inequality of *dist*:

$$\text{dist}(u_2, d_1) \leq w(u_1, u_2) + \text{dist}(u_1, d_1) \leq \varepsilon \Rightarrow u_2 \in \mathcal{B}(d_1, \varepsilon)$$

$$\Rightarrow \mathcal{B}(d_1, \varepsilon) \cap \mathcal{B}(d_2, \varepsilon) \neq \emptyset \Rightarrow D \notin \mathcal{D}(G, \varepsilon) \qquad \square$$

If we had to interpret the ε-consistency, let's say it's a weaker property than disjunction. It concerns paths directly connecting boxes without other vertices. Here is a characterization of Disjoint Complete Box Cover solutions:

Theorem 2.7. $\mathcal{C}(G, \varepsilon) \cap \mathcal{D}(G, \varepsilon) = \mathcal{C}(G, \varepsilon) \cap \mathcal{D}'(G, \varepsilon)$

Proof. $\mathcal{C}(G, \varepsilon) \cap \mathcal{D}(G, \varepsilon) \subset \mathcal{C}(G, \varepsilon) \cap \mathcal{D}'(G, \varepsilon)$ is the direct consequence of Th. 2.6.

For $\mathcal{C}(G, \varepsilon) \cap \mathcal{D}'(G, \varepsilon) \subset \mathcal{C}(G, \varepsilon) \cap \mathcal{D}(G, \varepsilon)$, let's consider $B \in \mathcal{C}(G, \varepsilon) \cap \mathcal{D}'(G, \varepsilon)$ and let's suppose that $B \notin \mathcal{D}(G, \varepsilon)$. That means that $\exists\ (b_1, b_2) \mid \mathcal{B}(b_1, \varepsilon) \cap \mathcal{B}(b_2, \varepsilon) \neq \emptyset$. Without loss of generality, that means that $\exists\ u_2 \in \mathcal{B}(b_2, \varepsilon) \mid \text{dist}(b_1, u_2) \leq \varepsilon$. Let's take the shortest path between b_1 and u_2. Let's call u_1 the last vertex of the path in $\mathcal{B}(b_1, \varepsilon)$. If u_1 is directly connected to u_2, that would violate the ε-consistency of b_1 and b_2. So that means u_1 is connected to a vertex $u_3 \notin \mathcal{B}(b_1, \varepsilon) \cup \mathcal{B}(b_2, \varepsilon)$. But $B \in \mathcal{C}(G, \varepsilon) \Rightarrow \exists\ b_3 \in B \mid u_3 \in \mathcal{B}(b_3, \varepsilon) \Rightarrow b_1$ and b_3 are not ε-consistent. So $B \notin \mathcal{D}'(G, \varepsilon)$. Contradiction. We illustrate this demonstration in Fig. 9. $\square$

This characterization enables us to construct every graph satisfying DCBC.

Proposition 2.7. *$\exists\ B \subseteq V \mid V = \bigcup_{b \in B} \mathcal{B}(b, \varepsilon)$ and E is compatible with the ε-consistency of $B \Leftrightarrow DCBC(G, \varepsilon) = true$*

In easier words, exactly every Disjoint Complete Box Covers can be constructed by connecting disjoint boxes with edges which doesn't violate the ε-consistency of B. We illustrate box cover partition considering the set $\mathcal{D}'(G, \varepsilon)$ in Fig. 10.

On another hand, we can wonder the following question: can we deduce a disjoint box cover from a complete box cover or vice versa?

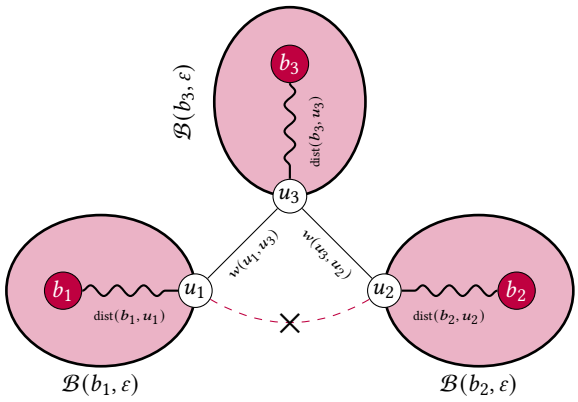

Fig. 9. The existence of the edge (u_1, u_2) violates B ε-consistency but the existence of the edge (u_1, u_3) too.

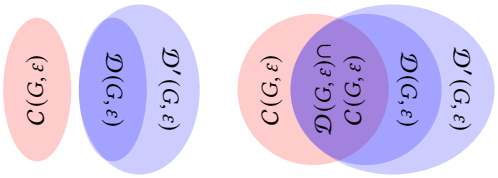

Fig. 10. On the left: ε-consistency box cover partition when disjoint and compact box covers intersection is empty. On the right: ε-consistency box cover partition when disjoint and compact box covers intersection isn't empty.

2.4 Conversion between DBC and CBC

Let's defined uncovered and overcovered vertices of disjoint and complete boxes. Uncovered vertices correspond to vertices not belonging to any disjoint box. Overcovered vertices correspond to vertices belonging to more than one complete box. Here are there respective sets:

$$\mathcal{U}(D, \varepsilon) = \left\{v \in V \mid \nexists\ d \in D \mid v \in \mathcal{B}(d, \varepsilon)\right\} = V \setminus (\biguplus_{d \in D} \mathcal{B}(d, \varepsilon))$$

$$= \{v \in V \mid |\mathcal{B}(v, \varepsilon) \cap D| < 1\} = V \setminus \{v \in V \mid |\mathcal{B}(v, \varepsilon) \cap D| = 1\}$$

$$\mathcal{O}(C, \varepsilon) = \left\{v \in V \mid \exists\ c \neq c' \in C \mid v \in \mathcal{B}(c, \varepsilon) \cap \mathcal{B}(c', \varepsilon)\right\}$$

$$= \{v \in V \mid |\mathcal{B}(v, \varepsilon) \cap C| > 1\} = V \setminus \{v \in V \mid |\mathcal{B}(v, \varepsilon) \cap C| = 1\}$$

Theorem 2.8. *If $DBC(G, \varepsilon) = CDC(G, \varepsilon)$, uncovered / overcovered vertices of optimal disjoint / complete box covers cannot be vertices of optimal complete / disjoint box covers.*

Proof. First, we can obviously notice that:

Lemma 2.9. $C \in \mathcal{C}(G, \varepsilon) \Leftrightarrow \mathcal{U}(C, \varepsilon) = \emptyset$

Lemma 2.10. $D \in \mathcal{D}(G, \varepsilon) \Leftrightarrow \mathcal{O}(D, \varepsilon) = \emptyset$

By knowing $\overline{f_{D,C}}$ is bijective $\forall\ (D, C) \in \overline{\mathcal{D}}(G, \varepsilon) \times \overline{\mathcal{C}}(G, \varepsilon)$ thanks to Th. 2.4, we can deduce that:

$$(\forall\ d \in D, \exists\,!\ c \in C \mid d \in \mathcal{B}(c, \varepsilon)) \wedge (\forall\ c \in C, \exists\,!\ d \in D \mid c \in \mathcal{B}(d, \varepsilon))$$

$$\Rightarrow (\forall\ d \in D, d \notin \mathcal{O}(C, \varepsilon)) \wedge (\forall\ c \in C, c \notin \mathcal{U}(D, \varepsilon))$$

$$\Leftrightarrow (D \cap \mathcal{O}(C, \varepsilon) = \emptyset) \wedge (C \cap \mathcal{U}(D, \varepsilon) = \emptyset)$$

That means that knowing a specific box cover gives us a lot of information on a box cover of the opposite problem. It will be very useful to create an conversion algorithm between box covers (see section 6.2). □

Let's denote uncovered vertices and overcovered vertices of a graph (and not a specific box cover) as following:

$$\mathcal{U}(\mathcal{D}(G,\varepsilon)) = \bigcup_{D\in\overline{\mathcal{D}}(G,\varepsilon)} \mathcal{U}(D,\varepsilon)\ ,\ \mathcal{O}(\mathcal{C}(G,\varepsilon)) = \bigcup_{C\in\overline{\mathcal{C}}(G,\varepsilon)} \mathcal{O}(C,\varepsilon)$$

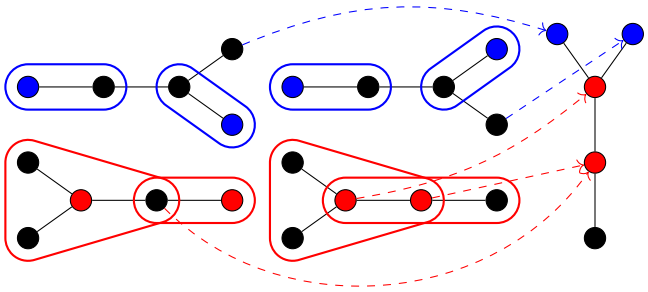

Fig. 11. On the upper left, every disjoint box cover in $\mathcal{D}(G,1)$. On the lower left, every complete box cover in $\mathcal{C}(G,1)$. On the right, $\mathcal{U}(\mathcal{D}(G,\varepsilon))$ vertices are colored in blue and $\mathcal{O}(\mathcal{C}(G,\varepsilon))$ vertices are colored in red.

Corollary 2.10.1. $\forall\ (D,C) \in \overline{\mathcal{D}}(G,\varepsilon) \times \overline{\mathcal{C}}(G,\varepsilon)$,

$$(D \cap \mathcal{O}(\mathcal{C}(G,\varepsilon)) = \emptyset) \wedge (C \cap \mathcal{U}(\mathcal{D}(G,\varepsilon)) = \emptyset)$$

Proof. Directly derived from Th. 2.8 and the new definitions.

That means that knowing all box covers gives us even more information on possible box covers of the opposite problem. In Fig. 11, we can verify the validity of Cor. 2.10.1 on the graph G' in Fig. 7. □

2.5 Dependency on ε

In order to better understand our box-covers, it can be interesting to consider $\mathrm{CBC}(G,\varepsilon)$ or $\mathrm{DBC}(G,\varepsilon)$ as a function of ε.

Theorem 2.11. *$CBC(G,\cdot)$ is a decreasing function.*

Proof. First, let's prove this lemma:

Lemma 2.12. $\epsilon_1 \leq \varepsilon_2 \Rightarrow \mathcal{C}(G,\varepsilon_1) \subseteq \mathcal{C}(G,\varepsilon_2)$

Proof. $\epsilon_1 \leq \varepsilon_2 \Rightarrow \forall\ u \in V,\ \mathcal{B}(u,\varepsilon_1) \subseteq \mathcal{B}(u,\varepsilon_2) \Rightarrow$

$$(\forall\ C \subseteq V,\ V = \textstyle\bigcup_{c\in C} \mathcal{B}(u,\varepsilon_1) \Rightarrow V = \bigcup_{c\in C} \mathcal{B}(u,\varepsilon_2))$$
$$\Rightarrow \mathcal{C}(G,\varepsilon_1) \subseteq \mathcal{C}(G,\varepsilon_2)$$ □

So we can now use it to conclude that if $\epsilon_1 \leq \varepsilon_2$:

$$C \in \overline{\mathcal{C}}(G,\varepsilon_1) \Leftrightarrow C \in \mathcal{C}(G,\varepsilon_1) \mid |C| = \mathrm{CBC}(G,\varepsilon_1)$$
$$\underset{\text{Lem. 2.12}}{\Rightarrow} C \in \mathcal{C}(G,\varepsilon_2) \mid |C| = \mathrm{CBC}(G,\varepsilon_1)$$
$$\Leftrightarrow \text{CBC-DEC}(G,\varepsilon_2,\mathrm{CBC}(G,\varepsilon_1)) = \text{true}$$
$$\underset{\text{Prop. 2.5}}{\Leftrightarrow} \mathrm{CBC}(G,\varepsilon_2) \leq \mathrm{CBC}(G,\varepsilon_1)$$ □

Theorem 2.13. *$DBC(G,\cdot)$ is a decreasing function.*

Proof. First, let's prove this lemma:

Lemma 2.14. $\epsilon_1 \leq \varepsilon_2 \Rightarrow \mathcal{D}(G,\varepsilon_2) \subseteq \mathcal{D}(G,\varepsilon_1)$

Proof. $\epsilon_1 \leq \varepsilon_2 \Rightarrow \forall\ u \in V,\ \mathcal{B}(u,\varepsilon_1) \subseteq \mathcal{B}(u,\varepsilon_2) \Rightarrow$

$$(\forall\ d \neq d' \in V, \mathcal{B}(d,\varepsilon_2) \cap \mathcal{B}(d',\varepsilon_2) = \emptyset \Rightarrow \mathcal{B}(d,\varepsilon_1) \cap \mathcal{B}(d',\varepsilon_1) = \emptyset)$$
$$\Rightarrow \mathcal{D}(G,\varepsilon_2) \subseteq \mathcal{D}(G,\varepsilon_1)$$ □

So we can now use it to conclude that if $\epsilon_1 \leq \varepsilon_2$:

$$D \in \overline{\mathcal{D}}(G,\varepsilon_2) \Leftrightarrow D \in \mathcal{D}(G,\varepsilon_2) \mid |D| = \mathrm{DBC}(G,\varepsilon_2)$$
$$\underset{\text{Lem. 2.14}}{\Rightarrow} D \in \mathcal{D}(G,\varepsilon_1) \mid |D| = \mathrm{DBC}(G,\varepsilon_2)$$
$$\Leftrightarrow \text{DBC-DEC}(G,\varepsilon_1,\mathrm{DBC}(G,\varepsilon_2)) = \text{true}$$
$$\underset{\text{Prop. 2.6}}{\Leftrightarrow} \mathrm{DBC}(G,\varepsilon_1) \geq \mathrm{DBC}(G,\varepsilon_2)$$ □

Just as a reminder, we've also got $\mathrm{CBC}(G,\cdot) \geq \mathrm{DBC}(G,\cdot)$ thanks to Th. 2.3.1. In Fig. 12, you can see an example of the $\mathrm{BC}(G,\cdot)$ graph associated with the graph G.

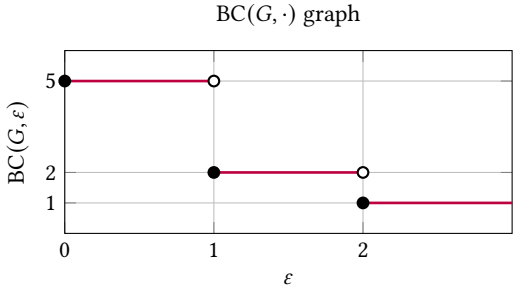

Fig. 12. The $\mathrm{BC}(G,\cdot)$ graph where G is the graph in Fig. 11 (in this specific case, $\mathrm{CBC}(G,\cdot) = \mathrm{DBC}(G,\cdot)$).

Theorem 2.15. *$CBC(G,\cdot)$ is a left-continuous step function.*

Proof. The fact that $\mathrm{CBC}(G,\cdot)$ is a step function follows directly the inclusion of its image set in $\mathbb{N}^*$. Proving that it is left-continuous is way more tricky.

Let's denote $\forall\ k \in [\![1,|V|]\!]$:

$$S_c(k) = \{\varepsilon \in \mathbb{R}_+ \mid \mathrm{CBC}(G,\varepsilon) = k\}$$

We're going to prove the left-continuity of $\mathrm{CBC}(G,\cdot)$ by demonstrating that $\forall\ k \in \mathrm{CBC}(G,\mathbb{R}_+)$ (it's the $\mathrm{CBC}(G,\cdot)$ image set), $\inf S_c(k)$ is reached. First, let's denote $\forall\ B \subseteq V$:

$$b_c(B) = \max_{u\in V} \min_{b\in B} \mathrm{dist}(b,u)$$

$b_c(B)$ corresponds to the maximal chemical distance from any vertex to its nearest box center of B.

Here is a characterization of the coverage of a given box cover thanks to b_c function.

Lemma 2.16. $b_c(C) \leq \varepsilon \Leftrightarrow C \in \mathcal{C}(G,\varepsilon)$

Proof. $\varepsilon < b_c(C) \Leftrightarrow \exists\, u \in V \mid \varepsilon < \min_{c\in C} \text{dist}(c,u)$

$\Leftrightarrow \exists\, u \in V \mid \forall c \in C,\ \varepsilon < \text{dist}(c,u) \Leftrightarrow u \notin \mathcal{B}(c,\varepsilon)$

$\Leftrightarrow \mathcal{U}(C,\varepsilon) \neq \emptyset \underset{\text{Lem. 2.9}}{\Leftrightarrow} C \notin \mathcal{C}(G,\varepsilon)$ □

To give an example, in Fig. 11, if we denote C the lower left complete box, $b_c(C) = 1$. We can also verify the validity of the coverage characterization thanks to the $\text{BC}(G,\cdot)$ graph in Fig. 12. Indeed, $\forall\, \varepsilon \geq 1,\ C \in \mathcal{C}(G,\varepsilon)$. Then, let's denote $\forall \varepsilon \in \mathbb{R}_+$:

$$b_c(\varepsilon) = \min_{C\in\overline{\mathcal{C}}(G,\varepsilon)} b_c(C)$$

Lemma 2.17. $\forall\, k \in [\![1,|V|]\!],\ \varepsilon \in S_c(k) \Rightarrow b_c(\varepsilon) \in [\inf S_c(k), \varepsilon]$

Proof. The definition of $b_c(\varepsilon)$ implies the existence of a box cover $C^* \in \overline{\mathcal{C}}(G,\varepsilon)$ such that $|C^*| = k$ and $b_c(C^*) = b_c(\varepsilon)$. Let's use this cover to prove the upper limit:

$$b_c(C^*) = b_c(\varepsilon) \text{ and } C^* \in \overline{\mathcal{C}}(G,\varepsilon) \underset{\text{Lem. 2.16}}{\Rightarrow} \varepsilon \geq b_c(\varepsilon)$$

Let's use this cover to prove the lower limit:

$$b_c(C^*) \leq b_c(\varepsilon) \underset{\text{Lem. 2.16}}{\Rightarrow} C^* \in \mathcal{C}(G, b_c(\varepsilon))$$

$$C^* \in \mathcal{C}(G, b_c(\varepsilon)) \text{ and } |C^*| = k \underset{\text{Prop. 2.5}}{\Rightarrow} \text{CBC}(G, b_c(\varepsilon)) \leq k$$

$$b_c(\varepsilon) \leq \varepsilon \underset{\text{Th. 2.11}}{\Rightarrow} \text{CBC}(G, b_c(\varepsilon)) \geq \text{CBC}(G,\varepsilon) = k$$

$$\text{CBC}(G, b_c(\varepsilon)) \leq k \text{ and } \text{CBC}(G, b_c(\varepsilon)) \geq k \Rightarrow$$

$$\text{CBC}(G, b_c(\varepsilon)) = k \Rightarrow b_c(\varepsilon) \geq \inf S_c(k)$$ □

Here again, in the same Fig. 11, $b_c(1.5) = 1$. We can also verify the validity of our order relationship thanks to the $\text{BC}(G,\cdot)$ graph in Fig. 12. Indeed, $1.5 \in S_c(2) \Rightarrow 1 \in [1,2]$.

Lemma 2.18. $\forall\, \varepsilon \in S_c(k), \varepsilon^* < b_c(\varepsilon) \Leftrightarrow \overline{\mathcal{C}}(G,\varepsilon) \cap \mathcal{C}(G,\varepsilon^*) = \emptyset$

Proof. $\varepsilon^* < b_c(\varepsilon) \Leftrightarrow \forall\, C \in \overline{\mathcal{C}}(G,\varepsilon),\ \varepsilon^* < b_c(C) \underset{\text{Lem. 2.16}}{\Leftrightarrow}$

$\forall\, C \in \overline{\mathcal{C}}(G,\varepsilon), C \notin \mathcal{C}(G,\varepsilon^*) \Leftrightarrow \overline{\mathcal{C}}(G,\varepsilon) \cap \mathcal{C}(G,\varepsilon^*) = \emptyset$ □

Finally, let's consider a sequence $(\varepsilon_i)_{i\in\mathbb{N}} \in S_c(k)^{\mathbb{N}}$. We initialize ε_0 with any ε in $S_c(k)$. Now, we suppose we got an ε_i ($i \in \mathbb{N}$). If $\inf S_c(k)$ is never reached, this means that $b_c(\varepsilon_i, k) > \inf S_c(k)$ so $(\inf S_c(k), b_c(\varepsilon_i)) \neq \emptyset$ (order guaranteed by Lem. 2.17). So we can choose ε_{i+1} in $(\inf S_c(k), b_c(\varepsilon_i))$ and continue the procedure with the next iteration.

Thanks to Lem. 2.18, we know that:

$$\forall\, i < j \in \mathbb{N}, \varepsilon_i > \varepsilon_j \Rightarrow \overline{\mathcal{C}}(G,\varepsilon_i) \cap \overline{\mathcal{C}}(G,\varepsilon_j) = \emptyset$$

And CBC always has a solution implies that:

$$\forall\, i \in \mathbb{N},\ \overline{\mathcal{C}}(G,\varepsilon_i) \neq \emptyset \Rightarrow |\overline{\mathcal{C}}(G,\varepsilon_i)| \geq 1$$

$$\Big|\bigcup_{1\leq i\leq n} \overline{\mathcal{C}}(G,\varepsilon_i)\Big| = \sum_{1\leq i\leq n} |\overline{\mathcal{C}}(G,\varepsilon_i)| \geq n \underset{n\to+\infty}{\to} = +\infty$$

$$\text{But } |\{C \subseteq V \mid |C| = k\}| \geq \Big|\bigcup_{1\leq i\leq n} \overline{\mathcal{C}}(G,\varepsilon_i)\Big|$$

And $\{C \subseteq V \mid |C| = k\}$ is finite. Contradiction. So $\inf S_c(k)$ is reached. □

Theorem 2.19. *$\text{DBC}(G,\cdot)$ id a left-continuous step function.*

Proof. The fact that $\text{DBC}(G,\cdot)$ is a step function follows directly the inclusion of its image set in $\mathbb{N}^*$. Proving that it is left-continuous is way more tricky.

Let's denote $\forall\, k \in [\![1,|V|]\!]$:

$$S_d(k) = \{\varepsilon \in \mathbb{R}_+ \mid \text{DBC}(G,\varepsilon) = k\}$$

We're going to prove the left-continuity of $\text{DBC}(G,\cdot)$ by demonstrating that $\forall\, k \in \text{DBC}(G,\mathbb{R}_+) \setminus \{1\}$ (it's the $\text{DBC}(G,\cdot)$ image set), $\sup S_d(k)$ is never reached. First, let's denote $\forall\, B \subseteq V$:

$$b_d(B) = \min_{u\in V} \min_{b\neq b'\in B} \max(\text{dist}(b,u), \text{dist}(b',u))$$

$b_d(B)$ corresponds to the minimal maximum between the chemical distance from any vertex to the box centers in every tuple of B^2.

Here is a characterization of the disjonction of a given box cover thanks to b_d function.

Lemma 2.20. $b_d(D) > \varepsilon \Leftrightarrow D \in \mathcal{D}(G,\varepsilon)$

Proof. $b_d(D) \leq \varepsilon \Leftrightarrow$

$$\exists\, u \in V \mid \varepsilon \geq \min_{d\neq d'\in D} \max(\text{dist}(d,u), \text{dist}(d',u))$$

$$\Leftrightarrow \exists\, u \in V, d \neq d' \in D \mid\ \varepsilon \geq \max(\text{dist}(d,u), \text{dist}(d',u))$$

$$\Leftrightarrow \exists\, u \in V,\ d \neq d' \in D \mid\ u \in \mathcal{B}(d,\varepsilon) \cap \mathcal{B}(d',\varepsilon)$$

$$\Leftrightarrow \mathcal{O}(D,\varepsilon) \neq \emptyset \underset{\text{Lem. 2.10}}{\Leftrightarrow} D \notin \mathcal{D}(G,\varepsilon)$$ □

To give an example, in Fig. 11, if we denote D the upper left disjoint box, $b_d(D) = 2$. We can also verify the validity of our disjunction characterization thanks to the $\text{BC}(G,\cdot)$ graph in Fig. 12. Indeed, $\forall\, \varepsilon < 2,\ D \in \mathcal{D}(G,\varepsilon)$. Then, let's denote $\forall \varepsilon \in \mathbb{R}_+$:

$$b_d(\varepsilon) = \max_{D\in\overline{\mathcal{D}}(G,\varepsilon)} b_d(D)$$

Lemma 2.21. $\forall\, k \in [\![2,|V|]\!],\ \varepsilon \in S_d(k) \Rightarrow \sup S_d(k) \leq b_d(\varepsilon)$

Proof. Let's use this cover to prove the upper limit. Let's take an $\varepsilon \in S_d(k)$ and let's suppose that $\text{DBC}(G, b_d(\varepsilon)) = k$. This implies that $\exists\, D^* \in \overline{\mathcal{D}}(G, b_d(\varepsilon)) \mid |D^*| = k$. But as:

$$\varepsilon < b_d(\varepsilon) \underset{\text{Lem. 2.14}}{\Rightarrow} D^* \in \mathcal{D}(G,\varepsilon)$$

$$|D^*| = k \Rightarrow D \in \overline{\mathcal{D}}(G,\varepsilon) \Rightarrow b_d(D^*) \leq b_d(\varepsilon)$$

$$D^* \in \mathcal{D}(G, b_d(\varepsilon)) \underset{\text{Lem. 2.20}}{\Rightarrow} b_d(D^*) > b_d(\varepsilon) \Rightarrow \text{ Contradiction.}$$

So $\text{DBC}(G, b_d(\varepsilon)) \neq k$. On the other hand:

$$b_d(\varepsilon) > \varepsilon \underset{\text{Th. 2.13}}{\Rightarrow} \text{DBC}(G, b_d(\varepsilon)) \leq \text{DBC}(G,\varepsilon) = k$$

$$\Rightarrow \text{DBC}(G, b_d(\varepsilon)) < k \underset{\text{Th. 2.13}}{\Rightarrow} b_d(\varepsilon) \geq \sup S_d(k)$$ □

Here again, in the same Fig. 11, $b_d(1) = 2$. We can also verify the validity of our order relationship thanks to the $\text{BC}(G,\cdot)$ graph in Fig. 12. Indeed, $1 \in S_d(2) \Rightarrow 2 \leq 2$. You can notice that we excluded $k = 1$ from our consideration in Lem. 2.21 because $\sup S_d(1) = +\infty$.

Lemma 2.22. $\forall\, \varepsilon \in S_d(k), \varepsilon^* < b_d(\varepsilon) \Rightarrow \varepsilon^* \leq \sup S_d(k)$

Proof.

$$\varepsilon^* < b_d(\varepsilon) \Rightarrow \exists\, D^* \in \overline{\mathcal{D}}(G,\varepsilon) \mid |D^*| = k \text{ and } \varepsilon^* < b_d(D^*)$$
$$\underset{\text{Lem. 2.20}}{\Rightarrow} \exists\, D^* \in \mathcal{D}(G,\varepsilon^*) \mid |D^*| = k \underset{\text{Th. 2.13}}{\Rightarrow} \varepsilon^* \leq \sup S_d(k) \qquad \square$$

Finally, for an $\varepsilon \in S_d(k)$, let's suppose that $\sup S_d(k) \neq b_d(\varepsilon)$. Let's take an $\varepsilon^* \in (\sup S_d(k), b_d(\varepsilon))$ (order guaranteed by Lem. 2.21). Thanks to Lem. 2.22, we know that $\varepsilon^* \leq \sup S_d(k)$ Contradiction.

So $\sup S_d(k) = b_d(\varepsilon)$ and we already proved in the demonstration of Lem. 2.21 that $b_d(\varepsilon) \notin S_d(k)$. So $\sup S_d(k)$ is never reached. $\square$

For a given graph G, we denote:

$$n_c = |\mathrm{CBC}(G,\mathbb{R}_+)| = |\{k \in [\![1,|V|]\!] \mid \exists\, \varepsilon \in \mathbb{R}_+ \mid \mathrm{CBC}(G,\varepsilon) = k\}|$$

$$n_d = |\mathrm{DBC}(G,\mathbb{R}_+)| = |\{k \in [\![1,|V|]\!] \mid \exists\, \varepsilon \in \mathbb{R}_+ \mid \mathrm{DBC}(G,\varepsilon) = k\}|$$

Now that we know that both $\mathrm{CBC}(G,\cdot)$ and $\mathrm{DBC}(G,\cdot)$ step functions are left-continuous, we can extract two sequences. We define $(\varepsilon_c^i)_{i\in[\![1,n_c]\!]} \in [\![1,|V|]\!]^{n_c}$ and $(\varepsilon_d^i)_{i\in[\![1,n_d]\!]} \in [\![1,|V|]\!]^{n_d}$ such as:

$$\varepsilon_c^i = \inf S_c(k_i) \text{ and } \varepsilon_d^i = \inf S_d(k_i)$$

where k_i is the i-th element of $\mathrm{CBC}(G,\mathbb{R}_+)$ or $\mathrm{DBC}(G,\mathbb{R}_+)$. These sequences corresponds to the left bounds of each step of $\mathrm{CBC}(G,\cdot)$ and $\mathrm{DBC}(G,\cdot)$. We can easily notice that $\varepsilon_c^{n_c} = \varepsilon_d^{n_d} = 1$ and that $\varepsilon_c^1 = \varepsilon_d^1 = |V|$. We can reconstruct the function from the sequence and the sequence from the function so these sequences contains as much information as the function. It is reasonable to assume that a linear regression of the box cover sizes to compute fractal dimension would be optimal if box covers were computed only in these epsilons. In Fig. 13, you can see an example of the $(\varepsilon_b^i)_{i\in[\![1,n_b]\!]}$ sequence associated with the $\mathrm{BC}(G,\cdot)$ graph of G.

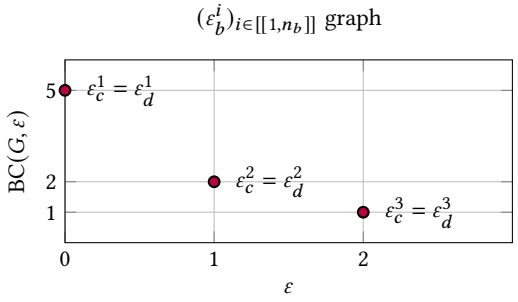

Fig. 13. The $(\varepsilon_b^i)_{i\in[\![1,n_b]\!]}$ sequence associated with the $\mathrm{BC}(G,\cdot)$ graph in Fig. 12 (in this specific case, $(\varepsilon_c^i)_{i\in[\![1,n_c]\!]} = (\varepsilon_d^i)_{i\in[\![1,n_d]\!]}$).

Theorem 2.23. $\varepsilon_c^i, \varepsilon_d^i \in \{dist(u,v) \mid u,v \in V\}$

Proof. In the demonstration of the theorem Th. 2.15, we proved that left bounds of $\mathrm{CBC}(G,\cdot)$ correspond to a given $b_c(\varepsilon)$. In the demonstration of the theorem Th. 2.19, we proved that right bounds of $\mathrm{DBC}(G,\cdot)$ (pretty much the same than left bounds) correspond to a given $b_d(\varepsilon)$. $b_c(\varepsilon)$ and $b_d(\varepsilon)$ corresponds to a given $b_c(C)$ and $b_d(D)$. And according to their definitions, $b_c(C), b_d(D) \in \{\mathrm{dist}(u,v) \mid u,v \in V\}$. $\square$

This theorem is a quite good result because it indicates us with which epsilon we should compute our box covers.

Here are two other interesting lemmas about b_c and b_d functions the followings ones:

Lemma 2.24. $D \in \overline{\mathcal{D}}(G,\varepsilon) \Rightarrow b_c(D) \leq 2\varepsilon$

Proof.
$$b_c(D) > 2\varepsilon \Leftrightarrow \exists\, u \in V \mid \forall\, d \in D,\ \mathrm{dist}(u,d) > 2\varepsilon$$
$$\Leftrightarrow \exists\, u \in V \mid \forall\, d \in D,\ (v \in \mathcal{B}(u,\varepsilon) \Rightarrow v \notin \mathcal{B}(d,\varepsilon))$$
$$\Leftrightarrow \exists\, u \in V \mid \mathcal{B}(u,\varepsilon) \cap \textstyle\bigcup_{d\in D} \mathcal{B}(d,\varepsilon) = \emptyset$$
$$\Leftrightarrow \exists\, u \in V \mid (D \in \mathcal{D}(G,\varepsilon) \Rightarrow D \cup \{u\} \in \mathcal{D}(G,\varepsilon))$$
$$\Rightarrow \exists\, D' \mid |D'| > |D| \text{ and } (D \in \mathcal{D}(G,\varepsilon) \Rightarrow D' \in \mathcal{D}(G,\varepsilon))$$
$$\Rightarrow D \notin \overline{\mathcal{D}}(G,\varepsilon) \qquad \square$$

It means that b_c function can be used both to characterize compact box covers and to identify unoptimal disjoint box covers.

Lemma 2.25. $C \in \overline{\mathcal{C}}(G,\varepsilon) \Rightarrow 2b_d(C) > \varepsilon$

Proof.
$$2b_d(C) \leq \varepsilon \Leftrightarrow$$
$$\exists\, u \in V, c \neq c' \in C \mid\ 2\max(\mathrm{dist}(u,c), \mathrm{dist}(u,c')) \leq \varepsilon$$
$$\Leftrightarrow \exists\, u \in V, c \neq c' \in C \mid\ (v \in \mathcal{B}(c,\varepsilon) \cup \mathcal{B}(c',\varepsilon) \Rightarrow v \in \mathcal{B}(u,\varepsilon))$$
$$\Leftrightarrow \exists\, u \in V, c \neq c' \in C \mid \mathcal{B}(c,\varepsilon) \cup \mathcal{B}(c',\varepsilon) \subseteq \mathcal{B}(u,\varepsilon)$$
$$\Leftrightarrow \exists\, u \in V \mid (C \in \mathcal{C}(G,\varepsilon) \Rightarrow C \cup \{u\} \setminus \{c,c'\} \in \mathcal{C}(G,\varepsilon))$$
$$\Rightarrow \exists\, C' \mid |C'| < |C| \text{ and } (C \in \mathcal{C}(G,\varepsilon) \Rightarrow C' \in \mathcal{C}(G,\varepsilon))$$
$$\Rightarrow C \notin \overline{\mathcal{C}}(G,\varepsilon) \qquad \square$$

It means that b_d function can be used both to characterize disjoint box covers and to identify unoptimal compact box covers.

These lemmas allows to conclude a very useful property:

Theorem 2.26. $\forall \varepsilon > 0, CBC(\cdot, 2\varepsilon) \leq DBC(\cdot,\varepsilon)$

Proof. $C \in \overline{\mathcal{C}}(G,2\varepsilon) \underset{\text{Lem. 2.25}}{\Rightarrow} 2b_d(C) > 2\varepsilon \Leftrightarrow b_d(C) > \varepsilon$
$$\underset{\text{Lem. 2.20}}{\Leftrightarrow} C \in \mathcal{D}(G,\varepsilon) \Rightarrow |C| \leq \mathrm{DBC}(G,\varepsilon)$$
$$\Rightarrow \mathrm{CBC}(G,2\varepsilon) \leq \mathrm{DBC}(G,\varepsilon) \qquad \square$$

We could have used lemmas Lem. 2.24 and Lem. 2.16 to obtain a very similar demonstration by using a $D \in \overline{\mathcal{D}}(G, \frac{\varepsilon}{2})$.

3 Complexity

3.1 $\mathcal{NP}$ nature of box cover decision problems

In this section, we will focus on the complexity of box cover problems.

Theorem 3.1. *CBC-DEC, DBC-DEC* $\in \mathcal{NP}$

Proof. Here are two algorithms called CBC-DEC verifier and DBC-DEC verifier.

Complexity of algorithms 1 and 2 is polynomial in input size (2 nested for loops over C and $|\mathcal{B}(c,\varepsilon)| \leq |V|$ means an $O(|C| \times |V|)$ complexity, same for D). So if CBC-DEC or DBC-DEC is true for given inputs, it exists a certificate C or D corresponding to the centers of the box cover that can prove it in polynomial time. This implies their belonging to $\mathcal{NP}$. $\square$

Algorithm 1 Verifier for CBC-DEC

Require: $G = (V, E)$, $\varepsilon \in \mathbb{R}$, $k \in \mathbb{N}$, $C \subseteq V$
Ensure: $B = (C \in \mathcal{C}(G, \varepsilon)) \wedge (|C| = k)$
  $B \leftarrow$ true, $P \leftarrow \emptyset$
  if $|C| \neq k$ **then**
    $B \leftarrow$ false
  end if
  for $c \in C$ **do**
    for $v \in \mathcal{B}(c, \varepsilon)$ **do**
      if $v \notin P$ **then**
        $P \leftarrow P \cup \{v\}$
      end if
    end for
  end for
  $B \leftarrow B \wedge (P = V)$

Algorithm 2 Verifier for DBC-DEC

Require: $G = (V, E)$, $\varepsilon \in \mathbb{R}$, $k \in \mathbb{N}$, $D \subseteq V$
Ensure: $B = (D \in \mathcal{D}(G, \varepsilon)) \wedge (|D| = k)$
  $B \leftarrow$ true, $P \leftarrow V$
  if $|D| \neq k$ **then**
    $B \leftarrow$ false
  end if
  for $d \in D$ **do**
    for $v \in \mathcal{B}(d, \varepsilon)$ **do**
      if $v \in P$ **then**
        $P \leftarrow P \setminus \{v\}$
      else
        $B \leftarrow$ false
      end if
    end for
  end for

3.2 $\mathcal{NP}$-completeness of box cover decision problems

Theorem 3.2. *CBC-DEC and DBC-DEC are $\mathcal{NP}$-complete.*

Proof. We have to reduce well-known $\mathcal{NP}$-complete problems to CBC-DEC and DBC-DEC. Here are SET-COVER and SET-PACKING decision problems. Some simple examples are provided in Fig.14.

Set Cover decision problem:

- Input: $U = [\![1, n]\!]$, $S \subseteq \mathcal{P}(U)$ such that $U = \bigcup_{s \in S} s$, $k \leq |S|$
- Output: Is there a subset set $S' \subseteq S$ such that $|S'| = k$ and $U = \bigcup_{s \in S} s$?

Set Packing decision problem:

- Input: $U = [\![1, n]\!]$, $S \subseteq \mathcal{P}(U)$, $k \leq |S|$
- Output: Is there a subset set $S' \subseteq S$ such that $|S'| = k$ and $(s)_{s \in S'}$ are pairwise disjoint?

These 2 problems have been proven $\mathcal{NP}$-complete by Richard Karp in 1972 (they belongs to Karp's 21 $\mathcal{NP}$-complete problems) [8]. In order to reduce box cover decision problems to set decision problems, we need to create a procedure to convert set decision instances to box cover decision instances. $\forall\, U = [\![1, n]\!]$, $S \subseteq \mathcal{P}(U)$,

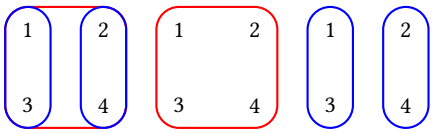

Fig. 14. a given $U = \{1, 2, 3, 4\}$, $S = \{\{1, 2, 3, 4\}, \{1, 2\}, \{3, 4\}\}$, a set cover corresponding to SET-COVER-DEC$(U, S, 1)$ and a set packing SET-PACKING-DEC$(U, S, 2)$

we define graphs $G_c(U, S)$ and $G_d(U, S)$:

$$G_c(U, S) = \begin{pmatrix} V = U \cup \{v_s \mid s \in S\}, \\ E = \{(u, v_s) \mid s \in S, u \in s\} \cup \{(v_s, v_{s'}) \mid s \neq s' \in S\} \end{pmatrix}$$

$$G_d(U, S) = \begin{pmatrix} V = U \cup \{v_s \mid s \in S\}, \\ E = \{(u, v_s) \mid s \in S, u \in s\} \cup \{(u, u') \mid u \neq u' \in U\} \end{pmatrix}$$

We added $|S|$ vertices v_s. This procedure is obviously polynomial in U and S sizes. An example is provided in Fig. 15 with set instances of Fig. 14.

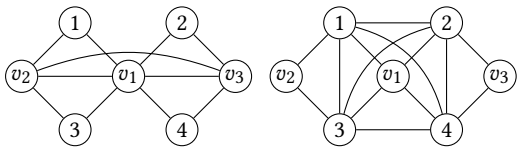

Fig. 15. $G_c(U, S)$ and $G_d(U, S)$ with U and S defined in Fig.14

Now, here are two new algorithms.

Algorithm 3 Reduction from SET-COVER-DEC to CBC-DEC

Require: $U = [\![1, n]\!]$, $S \subseteq \mathcal{P}(U)$ such that $U = \bigcup_{s \in S} s$, $k \leq |S|$
Ensure: $B =$ SET-COVER-DEC(U, S, k)
  $G = G_c(U, S)$
  $B =$ CBC-DEC$(G, 1, k)$

Algorithm 4 Reduction from SET-PACKING-DEC to DBC-DEC

Require: $U = [\![1, n]\!]$, $S \subseteq \mathcal{P}(U)$, $k \leq |S|$
Ensure: $B =$ SET-PACKING-DEC(U, S, k)
  $G = G_d(U, S)$
  $B =$ DBC-DEC$(G, 1, k)$

Theorem 3.3. *Algorithms 3 and 4 are reductions of SET-COVER-DEC and SET-PACKING-DEC to CBC-DEC and DBC-DEC.*

Proof. Let's call $V_S = \{v_s \mid s \in S\}$. As an example, $V_{G_c(U,S)} = V_{G_d(U,S)} = U \cup V_S$.
First, let's prove the following lemmas:

Lemma 3.4. *CBC-DEC$(G_c(U, S), 1, k)$ = true* $\Leftrightarrow$

$$\exists\, V \subseteq V_S \mid (|V| = k) \wedge (V \in \mathcal{C}(G_c(U, S), 1))$$

Proof. If we suppose CBC-DEC$(G_c(U,S),1,k)$ is true, we can take $C \in \mathcal{C}(G_c(U,S),1) \mid |C| = k$ and suppose that $\exists\, u \in C \mid u \in U$. u is connected only to vertices in V_S. Let's choose a vertex v_{s*} in these vertices. v_{s*} is connected to u and to the other vertices connected to u. So $C' = C \setminus \{u\} \cup \{v_{s*}\}$ covers exactly the same vertices than C so $C' \in \mathcal{C}(G_c(U,S),1)$ and $|C'| = |C| = k$. By repeating this process until there is no more $u \in U$ in our complete box cover, we obtain a set $V \subseteq (V_{G_c(U,D)} \setminus U) = V_S \mid (|V| = k) \wedge (V \in \mathcal{C}(G_c(U,S),1))$. To conclude, if CBC-DEC$(G_c(U,S),1,k)$ is true, it always exists $V \subseteq V_S \cap \mathcal{C}(G_c(U,S),1) \mid |V| = k$. The reciprocal is trivial. □

Lemma 3.5. *DBC-DEC*$(G_d(U,S),1,k) = true \Leftrightarrow$

$$\exists\, V \subseteq V_S \mid (|V| = k) \wedge (V \in \mathcal{D}(G_d(U,S),1))$$

Proof. If we suppose DBC-DEC$(G_d(U,S),1,k)$ is true, we can take $D \in \mathcal{D}(G_d(U,S),1) \mid |D| = k$ and suppose that $\exists\, u \in D \mid u \in U$. u is connected to every other vertex $u' \in U$ and every $v_s \in V_S$ is connected to a vertex of U so if we put a disjoint box centered on u in D, we can't have another vertex v_s in D (their boxes would intersect with $\mathcal{B}(u,1)$). We can conclude that $D \cap U \neq \emptyset \Rightarrow |D| = 1$. So if $k > 1$, D only contains vertices of V_S. And if $k = 1$, any box centered on any vertex of V_S would belong to $\mathcal{D}(G_d(U,S),1)$. To conclude, if DBC-DEC$(G_d(U,S),1,k)$ is true, it always exists $V \subseteq V_S \cap \mathcal{D}(G_d(U,S),1) \mid |V| = k$. The reciprocal is trivial. □

To be honest, we added new edges between vertices v_s in $G_c(U,S)$ and new edges between vertices u in $G_d(U,S)$ just to guarantee the validity of Lem. 3.4 and Lem. 3.5. Now, let's consider the following function $g : \mathcal{P}(S) \to \mathcal{P}(V_S)$:

$$\forall\, S' \subseteq S,\ g(S') = V_{S'}$$

Lemma 3.6. *Function g is bijective.*

Proof. We can demonstrate it by giving its inverse function:

$$\forall\, V \subseteq V_S,\ g^{-1}(V) = \left\{\{u \in U \mid (u,v) \in E_{G_c(U,S)}\} \mid v \in V\right\}$$
$$= \{\mathcal{B}(v,1) \setminus \{v\} \mid v \in V\}$$ □

Thanks to this bijective property, we can prove two more equivalences properties between coverage and disjunction of sets and box covers.

Lemma 3.7. $\exists\, S' \subseteq S \mid U = \bigcup_{s \in S'} s$

$$\Leftrightarrow \exists\, V \subseteq V_S \mid V \in \mathcal{C}(G_c(U,S),1)$$

Proof.

$$\exists\, S' \subseteq S \mid U = \bigcup_{s \in S'} s$$
$$\Leftrightarrow \exists\, S' \subseteq S \mid V_{G_c(U,S)} = \bigcup_{v \in g(S')} \mathcal{B}(v,1)$$
$$\Leftrightarrow \exists\, S' \subseteq S \mid g(S') \in \mathcal{C}(G_c(U,S),1)$$
$$\underset{\text{Lem. 3.6}}{\Leftrightarrow} \exists\, V \subseteq V_S \mid V \in \mathcal{C}(G_c(U,S),1)$$

□

Lemma 3.8. $\exists\, S' \subseteq S \mid (s)_{s \in S'}$ *are pairwise disjoint.*

$$\Leftrightarrow \exists\, V \subseteq V_S \mid V \in \mathcal{D}(G_d(U,S),1)$$

Proof.

$$\exists\, S' \subseteq S \mid (s)_{s \in S'} \text{ are pairwise disjoint.}$$
$$\Leftrightarrow \exists\, S' \subseteq S \mid (\mathcal{B}(v,1))_{v \in g(S')} \text{ are pairwise disjoint.}$$
$$\Leftrightarrow \exists\, S' \subseteq S \mid g(S') \in \mathcal{D}(G_d(U,S),1)$$
$$\underset{\text{Lem. 3.6}}{\Leftrightarrow} \exists\, V \subseteq V_S \mid V \in \mathcal{D}(G_d(U,S),1)$$

□

So we can conclude this equivalence for SET-COVER-DEC and CBC-DEC:

$$\text{SET-COVER-DEC}(U,S,k) = \text{true}$$
$$\Leftrightarrow \exists\, S' \subseteq S \mid (|S| = k) \ \wedge\ (U = \textstyle\bigcup_{s \in S'} s)$$
$$\underset{\text{Lem. 3.7}}{\Leftrightarrow} \exists\, V \subseteq V_S \mid (|V| = k) \wedge (V \in \mathcal{C}(G_c(U,S),1))$$
$$\underset{\text{Lem. 3.4}}{\Leftrightarrow} \text{CBC-DEC}(G_c(U,S),1,k) = \text{true}$$

And this equivalence for SET-PACKING-DEC and DCB-DEC:

$$\text{SET-PACKING-DEC}(U,S,k) = \text{true}$$
$$\Leftrightarrow \exists\, S' \subseteq S \mid (|S| = k) \ \wedge\ ((s)_{s \in S'} \text{ are pairwise disjoint.})$$
$$\underset{\text{Lem. 3.8}}{\Leftrightarrow} \exists\, V \subseteq V_S \mid (|V| = k) \wedge (V \in \mathcal{D}(G_d(U,S),1))$$
$$\underset{\text{Lem. 3.5}}{\Leftrightarrow} \text{DBC-DEC}(G_d(U,S),1,k) = \text{true}$$

We just proved that the equivalences between these two problems so reductions are correct. Fig. 16 shows CBC-DEC and DBC-DEC solutions for corresponding SET-COVER-DEC and SET-PACKING-DEC solutions in Fig. 14 (with graph instances shown in Fig. 15). □

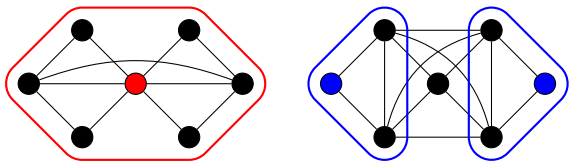

Fig. 16. Box covers corresponding to CBC-DEC$(G_c(U,S),1)$ and DBC-DEC$(G_d(U,S),2)$ with U and S defined in Fig.14

So Th. 3.3 allows us to conclude:

$$\text{SET-COVER-DEC} \leq \text{CBC-DEC}$$

$$\text{SET-PACKING-DEC} \leq \text{DBC-DEC}$$

So the $\mathcal{NP}$-completeness of SET-COVER and SET-PACKING decision problems implies the $\mathcal{NP}$-completeness of CBC-DEC and DBC-DEC.

Little observation: reduction from CBC-DEC and DBC-DEC to SET-COVER and SET-PACKING would be very simple, we would just have to set U as V and S as $\{\sqsubseteq, \varepsilon \mid v \in V\}$. □

Algorithm 5 Reduction from CBC-DEC to CBC

Require: $G = (V, E)$, $\varepsilon \in \mathbb{R}$, $k \in \mathbb{N}$
Ensure: $B = \text{CBC-DEC}(G, \varepsilon, k)$
$B = (\text{CBC}(G, \varepsilon) \leq k)$

Algorithm 6 Reduction from DBC-DEC to DBC

Require: $G = (V, E)$, $\varepsilon \in \mathbb{R}$, $k \in \mathbb{N}$
Ensure: $B = \text{DBC-DEC}(G, \varepsilon, k)$
$B = (\text{DBC}(G, \varepsilon) \geq k)$

3.3 $\mathcal{NP}$-hardness of box cover decision problems

Theorem 3.9. *CBC and DBC are $\mathcal{NP}$-hard.*

Proof. Here are 2 obvious reductions from CBC-DEC and DBC-DEC to CBC and DBC justified by Prop. 2.5 and Prop.2.6.
So we can conclude:

$$\text{CBC-DEC} \leq \text{CBC and DBC-DEC} \leq \text{DBC}$$

So CBC-DEC and DBC-DEC $\mathcal{NP}$-completeness implies CBC and DBC $\mathcal{NP}$-hardness. □

As a conclusion, we proved the $\mathcal{NP}$-complexity of box covers problems. We don't expect to find polynomial algorithms to solve it but we can try to find approximations based on greedy heuristics (cf section 6).

4 Operational Research

4.1 Integer Linear Programs

4.1.1 Box Cover Problem Linear Program formulation. In this two integer linear programs for Complete and Disjoint Box Cover, we have to number the vertices between 1 and $n = |V|$. b_i (either c_i or d_i depending on the problem) and $\varepsilon_{i,j}$ are defined as follows:

$$\forall\, i \in [\![1, n]\!],\ b_i = \begin{cases} 1 & \text{if vertex } v_i \text{ is the center of a box} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall\, i, j \in [\![1, n]\!],\ \varepsilon_{i,j} = \begin{cases} 1 & \text{if } \text{dist}(v_i, v_j) \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

We can now formulate Integer Linear Programs of CBC and DBC as follows:

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{n} c_i \\ &\text{subject to} && \sum_{i=1}^{n} \varepsilon_{i,j} \times c_i \geq 1 \quad \forall\, j \in [\![1, n]\!] \\ & && c_i \in \{0, 1\} \quad \forall\, i \in [\![1, n]\!] \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{n} d_i \\ &\text{subject to} && \sum_{i=1}^{n} \varepsilon_{i,j} \times d_i \leq 1 \quad \forall\, j \in [\![1, n]\!] \\ & && d_i \in \{0, 1\} \quad \forall\, i \in [\![1, n]\!] \end{aligned} \tag{2}$$

Proof. If we denote the set of box cover centers $B = \{v_i \mid b_i = 1\}$ (either C or D depending on the problem):

$$\forall\, i, j \in [\![1, n]\!],\ (\varepsilon_{i,j} \times b_i = 1) \Leftrightarrow \left(v_i \in B \wedge v_j \in \mathcal{B}(v_i, \varepsilon)\right)$$

$$\Rightarrow \forall j \in [\![1, n]\!],\ \sum_i \varepsilon_{i,j} \times b_i = \left|\left\{b \in B \mid v_j \in \mathcal{B}(b, \varepsilon)\right\}\right|$$

So we can conclude that:

$$\forall j \in [\![1, n]\!],\ \sum_i \varepsilon_{i,j} \times c_i \geq 1 \underset{\text{Lem. 2.10}}{\Leftrightarrow} C \in \mathcal{C}(G, \varepsilon)$$

$$\forall j \in [\![1, n]\!],\ \sum_i \varepsilon_{i,j} \times d_i \leq 1 \underset{\text{Lem. 2.9}}{\Leftrightarrow} D \in \mathcal{D}(G, \varepsilon)$$

b_i are the variables and $\varepsilon_{i,j}$ are deterministic factors (thanks to Dijkstra chemical distances computed before the linear program) so it's still a *linear* program. □

4.1.2 Box Cover Decision Problem Linear Program formulation. If we wanted to put these integer linear programs into standard form (i.e. constraints are equalities instead of inequalities), we would have to add to Eq.1 variables $x_j \in \mathbb{N}$ and to Eq.2 variables $y_j \in \mathbb{N}$ such that:

$$\sum_i \varepsilon_{i,j} \times c_i - x_j = 1 \text{ and } \sum_i \varepsilon_{i,j} \times d_i + y_j = 1 \quad \forall j$$

In practice, x_j corresponds to the number of complete balls the vertex v_j belongs to minus 1 ($x_j \in [\![0, |V| - 1]\!]$) and y_j corresponds to 1 minus the number of disjoint balls the vertex v_j belongs to ($y_j \in [\![0, 1]\!]$).

These integer linear programs also work for box cover decision problems. We just need to replace the optimization goal with one additional constraint in each program:

$$\sum_i c_i = k \text{ and } \sum_i d_i = k$$

So these integer linear program formulations ensures us that:

$$\begin{matrix} \text{CBC-DEC, CBC,} \\ \text{DBC-DEC, DBC} \end{matrix} \quad \leq \quad \text{INTEGER-LINEAR-PROGRAMS}$$

Since the decision problem associated with linear program optimization problem ("is there a feasible solution that meets the constraints?") is in $\mathcal{NP}$, we found another way to prove Th. 3.1.

4.2 Duality between DBC and CBC

Let's find the matrix forms of these Linear Programs. We can denote $B \in \mathcal{M}_{n,1}(\{0, 1\})$ the column matrix of complete box centers, $\mathbb{1} \in \mathcal{M}_{n,1}(\{0, 1\})$ the unitary column matrix and $E \in \mathcal{M}_n(\{0, 1\})$ the matrix of box belonging such that:

$$\forall\, i, j \in [\![1, n]\!],\ B_{i,1} = b_i,\ \mathbb{1}_{i,1} = 1,\ E_{i,j} = \varepsilon_{i,j}$$

We can notice that $E \in \mathcal{S}_n(\{0, 1\})$ thanks to *dist* symmetry. E matrix can be obtained from the chemical distance matrix D of the graph G with this relation (we generalize $\max(\cdot)$ notation to matrices):

Theorem 4.1. $E = \max(0, 1 - \max(0, \frac{D}{\varepsilon} - 1))$

Proof. $\forall\ i,j \in [[0,|V|-1]],\ E_{i,j} = 1 \Leftrightarrow$

$$\text{dist}(v_i,v_j) \leq \varepsilon \Leftrightarrow \frac{D_{i,j}}{\varepsilon} - 1 \leq 0 \Rightarrow \max(0, \frac{D_{i,j}}{\varepsilon} - 1) = 0$$
$$\Rightarrow \max(0, 1 - \max(0, \frac{D_{i,j}}{\varepsilon} - 1)) = 1$$
$$E_{i,j} = 0 \Leftrightarrow \text{dist}(v_i,v_j) > \varepsilon \Leftrightarrow \frac{D_{i,j}}{\varepsilon} - 1 > 0$$
$$\Rightarrow \max(0, \frac{D_{i,j}}{\varepsilon} - 1) = \frac{D_{i,j}}{\varepsilon} - 1$$
$$\Leftrightarrow 1 - \max(0, \frac{D_{i,j}}{\varepsilon} - 1) = -\frac{D_{i,j}}{\varepsilon} < 0$$
$$\Rightarrow \max(0, 1 - \max(0, \tfrac{D_{i,j}}{\varepsilon} - 1)) = 0 \qquad \square$$

Since $\forall\ i,j \in [[1,n]],\ \varepsilon_{i,j} = \varepsilon_{j,i}$, we can deduce $E \in \mathcal{S}_n(\{0,1\})$. We can now formulate the Linear Programs as follows:

$$\begin{aligned} &\text{minimize} && \mathbb{1}^T \times C \\ &\text{subject to} && E \times C \geq \mathbb{1} \\ & && C \in \mathcal{M}_{n,1}(\{0,1\}) \end{aligned} \tag{3}$$

$$\begin{aligned} &\text{maximize} && \mathbb{1}^T \times D \\ &\text{subject to} && E \times D \leq \mathbb{1} \\ & && D \in \mathcal{M}_{n,1}(\{0,1\}) \end{aligned} \tag{4}$$

Theorem 4.2. *CBC and DBC linear programs are duals.*

Proof. If we denote $D' = D^T$ and $\mathbb{1}' = \mathbb{1}^T$ can observe that:

$$E \times D \leq \mathbb{1} \Leftrightarrow D^T \times E^T \leq \mathbb{1}^T \Leftrightarrow D' \times E \leq \mathbb{1}'$$
$$\mathbb{1}^T \times D = D^T \times \mathbb{1} = D' \times \mathbb{1}'^T$$

So it appears obvious that DBC linear program is the dual of CBC linear program. $\square$

Since feasible solutions of DBC linear program are all in $\mathcal{D}(G,\varepsilon)$ and feasible solutions of CBC linear program are all in $\mathcal{C}(G,\varepsilon)$, using the weak duality theorem (the strong duality theorem isn't valid for integer linear programs), we found another way to prove Cor. 2.3.1.

4.3 Integer polyhedron

Here are some reminders about integer polyhedra:

$$\forall\ a \in \mathbb{R}^n,\ b \in \mathbb{R},\ \left\{x \in \mathbb{R}^n \mid a^T \times x \leq b\right\} \text{ is called a half-space.}$$

A polyhedron is the intersection of a finite number of half-spaces. If we denote A the constraint matrix of a given linear program (each row is a different a^T), we can obtain the polyhedron P associated to a linear program:

$$P = \left\{x \in \mathbb{R}^n \mid A \times x \leq b\right\}$$

$\forall\ c \in \mathbb{R}^n,\ \delta \in \mathbb{R},$ the inequality $c^T \times x \leq \delta$ is valid for P if $\forall\ x \in P,\ c^T \times x \leq \delta$. We call $v \in P$ a vertex of P if:

$$\exists\ c^T \times x \leq \delta \text{ valid for } P \mid \{x \in \mathbb{R}^n \mid c^T \times x \leq \delta\} = \{v\}$$

An integer polyhedron is a polyhedron whose vertices are integer. The linear relaxation of an integer linear program is the linear program obtained by replacing the $x \in \mathbb{Z}^n$ condition by $x \in \mathbb{R}^n$. It's proved that the solution of an integer linear program directly corresponds to the solution of its linear relaxation if its associated polyhedron is an integer polyhedron [4]. These is quite interesting because it is then possible to solve the associated integer linear programs in polynomial time (since it is sufficient to solve the linear programs corresponding to their linear relaxation). To date, there is no precise characterization of integer polyhedra, but an important result concerns totally unimodular matrices.

A matrix $A \in \mathcal{M}_{n,m}(\mathbb{Z})$ is totally unimodular if and only if:

$$\forall\ B \text{ sub-matrix of } A,\ \det(B) \in \{+1, 0, -1\}$$

It's proved that every unimodular matrices corresponds to integer polyhedra (it's a sufficient condition but not a necessary one) [7]. So we'd like to know when the E matrix of our linear programs is totally unimodular. In order to do that, we can consider E as the adjacency matrix of a given graph $G_E = (V_E, E_E)$.

$$E_E = \{(u,v) \in V^2 \mid u \in \mathcal{B}(v,\varepsilon)\}$$

An unimodular graph is a graph whose adjacency matrix is totally unimodular. It's also proved [1] that unimodular graphs are graphs whose adjacency matrix can be written as:

$$A = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \text{ where } B \text{ is totally unimodular}$$

That implies a bipartite partition of V_E. But unfortunately, $\forall\ v \in V_E$:

$$\text{dist}(v,v) = 0 \Leftrightarrow \forall\ \varepsilon \in \mathbb{R},\ v \in \mathcal{B}(v,\varepsilon) \Leftrightarrow (v,v) \in E_E$$

Vertices are always connected to themselves in G_E so bipartite partition cannot exist (and therefore E cannot be totally unimodular so we haven't any result on the integral polyhedra).

5 Variants

Box Covers problems have variants we will not necessarily study at great length in this paper. We'll mention four of the most important variants (but only three of them do not contradict the fundamentals of this paper). Be careful, in the first section, we assign weight to boxes, not to edges.

5.1 Weighted box cover

5.1.1 Principle. In weighted box cover problems, each box is assigned a positive weight (representing its cost) and each box cover is assigned the sum of box weights. The goal is to find a weighted complete box cover with a smallest weight or a weighted disjoint box cover with the greatest weight. Usually, we can assign to a box cover the sum of weights of edges used in shortest paths between its center and its vertices or just the number of such edges if there is not weights (we consider each edge cost 1).

For example, in Fig.17, $|C| = |C'| = 2$ but C weight cost is $3+3 = 6$ and C' weight cost is $2+3 = 5 < 6$. In the same way, $|D| = |D'| = 1$ but D weight cost is 2 and D' weight cost is $3 > 2$.

5.1.2 Applications. In practice, it makes sense. If we want to cover a telephone network with antennae, rather than just the number of antennae needed to cover all the homes, the cost of the installations (directly linked to the quantity of electrical cables, which in our graphs corresponds to the weight of the edges in the shortest paths) also matters. To be even more realistic, given the fact antennae are more expensive than connections, we could even seek first to

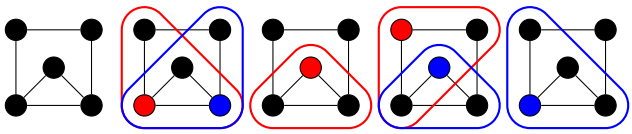

Fig. 17. a given graph G, box covers C and D corresponding to $\mathrm{CBC}(G,1)$, $\mathrm{DBC}(G,1)$ and optimal weight box covers C' and D'

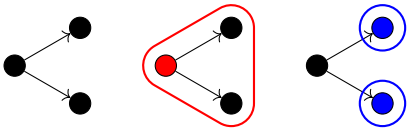

Fig. 19. A given directed graph G and two box covers corresponding to $\mathrm{CBC}(G,1)$ and $\mathrm{DBC}(G,1)$

minimize the number of antennas and then, by fixing the number of antennas using the previous result, minimize the number of connections.

5.2 Fractional box cover

5.2.1 Principle. In fractional box cover problems, it is allowed to select fractions of boxes, rather than entire boxes. A fractional box cover is an assignment of a fraction to each box in box covers, such that for each $v \in V$, the sum of fractions of boxes that contain v is at least 1 for complete boxes and at most 1 for disjoint box. The goal is to find a fractional complete box cover in which the sum of fractions is as small as possible or a fractional disjoint box cover in which the sum of fractions is as great as possible. Usual box covers are fractional box covers in which fractions are either 0 or 1. Therefore, smallest fractional complete box cover size is at most equal to smallest complete box cover size, but may be smaller. And greater fractional disjoint box cover size is at least equal to the greater disjoint box cover size, but may be greater.

For example, in Fig.18, $|C| = 2$ and $|C'| = 4$ but C fractional cost is 2 and C' fractional cost is $\frac{1}{3} \times 4 = \frac{4}{3} < 2$. In the same way, $|D| = 1$ and $|D'| = 4$ but D fractional cost is 1 and D' weight cost is $\frac{4}{3} > 1$.

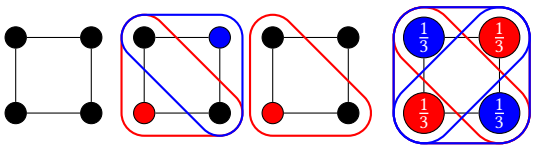

Fig. 18. a given graph G, box covers C and D corresponding to $\mathrm{CBC}(G,1)$, $\mathrm{DBC}(G,1)$ and an optimal fractional box cover (the same for both problem) $C' = D'$ (all vertices are in the box cover with fraction equal to $\frac{1}{3}$)

5.2.2 Applications. Here again, in practice, it makes sense. A given home doesn't need to receive all connection by only one antenna. It can receive different parts of its connection by different antennae.

5.3 Oriented box cover

5.3.1 Principle. In oriented box cover problems, we consider having oriented graphs instead of non-oriented ones. Problem with this one is that it contradicts the fundamental property Prop. 2.1 on which most demonstrations are based. Consequently, we can have disjoint box covers whose sizes are greater than complete box covers like shown in Fig. 19 which contradicts another fundamental theorem Th. 2.1 and oriented box cover become way more complicated to study.

5.3.2 Applications. But to be honest, in practice, it can make sense. Getting a product from point A to point B isn't necessarily as expensive as getting it from point B to point A. If you're moving water, getting it down a mountain is easier than getting it up. One of the reason we will not study these 3 problems is that although they have useful practical applications, they are useless for fractal dimension computation (the main goal of this paper).

5.4 Infinite graphs box covers

5.4.1 Principle. In Section 7.1, we're going to use box cover on infinite graphs. Please be aware that for all infinite graphs G, if $\mathrm{CBC}(G,\cdot)$ and $\mathrm{DBC}(G,\cdot)$ are well defined, we can easily show that the results proved before are still true. That's why we'll sometimes use theorems proved for finite graphs on infinite graphs without any justification.

5.4.2 Applications. Applications for infinite graphs box covers are less straightforward. We can consider very large graphs as infinite graphs but there are also applications in machine learning and neural networks. In convolutional neural networks, it can be applied to analyze and optimize convolutional receivers over infinite or very large spaces, which could have implications for network architectures and learning.

5.5 Linear Program formulations

In order to adapt linear programs we mentioned earlier in section 4.1 to our variants, we need to make very few changes.

For weighted box covers, we need to change the optimization goal. It will become:

$$\text{minimize} \quad \sum_i w_i \times c_i \quad \text{and} \quad \text{maximize} \quad \sum_i w_i \times d_i$$

Where the variables w_i are either arbitrarily chosen or, if we consider box cover weights are shortest past edges weights sums, computed before the linear program formulation. They are deterministic factors so it's still a *linear* program.

For fractional box covers, we need to change the variable domain of c_i and d_i. It will become:

$$0 \leq c_i \leq 1 \quad \forall i \quad \text{and} \quad 0 \leq d_i \leq 1 \quad \forall i$$

So with this formulation, integer linear program become linear program, a $\mathcal{P}$ problem. This ensures that fractional box cover is a $\mathcal{P}$ problem.

For oriented box cover, the formulation is identical as the oriented integer linear program but we need to consider that $\varepsilon_{i,j} \neq \varepsilon_{j,i}$ because $\mathrm{dist}(v_i, v_j) \neq \mathrm{dist}(v_j, v_i)$ so E is not more in $\mathcal{S}_n(\{0,1\})$. So Th. 4.2 is no more valid and this is another way to explain why we

have disjoint box covers whose sizes are greater than complete box covers.

For infinite graphs box covers, the integer linear program optimization become countable infinite integer linear program (which is a even harder problem) :

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{+\infty} c_i \\ &\text{subject to} && \sum_{i=1}^{+\infty} \varepsilon_{i,j} \times c_i \geq 1 \quad \forall\, j \geq 1 \\ & && c_i \in \{0,1\} \quad \forall\, i \geq 1 \end{aligned} \tag{5}$$

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{+\infty} d_i \\ &\text{subject to} && \sum_{i=1}^{+\infty} \varepsilon_{i,j} \times d_i \leq 1 \quad \forall\, j \geq 1 \\ & && d_i \in \{0,1\} \quad \forall\, i \geq 1 \end{aligned} \tag{6}$$

6 Approximations

As a reminder, an ρ-approximation $\mathcal{A}_{\mathcal{P}}$ of a minimization problem $\mathcal{P}$ is an algorithm such that:

$$\forall\, I \in \mathcal{I}, \frac{f(\mathcal{A}_{\mathcal{P}}(I))}{f(OPT_{\mathcal{P}}(I))} \leq \rho$$

Where $\mathcal{I}$ are the instances of the problem (these are the graphs and the ε), f the valuation function we want to minimize (it's the size of the box covers) and $OPT_{\mathcal{P}}$ the optimal solution we're trying to approximate. In a similar way, an ρ-approximation $\mathcal{A}$ of a maximization problem $\mathcal{P}$ is an algorithm such that:

$$\forall\, I \in \mathcal{I}, \frac{f(OPT_{\mathcal{P}}(I))}{f(\mathcal{A}_{\mathcal{P}}(I))} \leq \rho$$

An algorithm $\mathcal{A}_{\mathcal{P}}$ is an approximation of a problem $\mathcal{P}$ if and only if it exists $\rho \in \mathbb{R}$ such that $\mathcal{A}_{\mathcal{P}}$ is an ρ-approximation of $\mathcal{P}$.

6.1 Box cover creation

6.1.1 *Algorithms.*

Order relationship on vertices. In these two next algorithms created to solve CBC and DBC, we use two different heuristics based on a relationship order between vertices. If we name the degree of a given vertex $\deg_V(v)$ such as:

$$\forall\, v \in V,\ \deg_V(v) = \left|\{u \in V \mid (u,v) \in E\}\right| = (\left|(\{v\} \times V) \cap E\right|)$$

We define the order relationship $\preceq_V$ as follows:

$$\forall\, u, v \in V,\ u \preceq_V v \Rightarrow \deg_V(u) \leq \deg_V(v)$$

BC Heuristics. The CBC heuristic consists in picking as centers of complete box covers the maximum vertices according to $\preceq_V$. The assumption is that we need to minimize the numbers of boxes so we need to pick boxes "as big as possible". In reverse, the DBC heuristic consists in picking as center of disjoint box covers the minimum vertices according to $\preceq_V$. The assumption is that we need to maximize the number of boxes so we need to pick boxes "as small as possible".

The idea that degree distribution is linked with box covers doesn't come out of nowhere. Indeed, Barabási et. al [2] proved in 1999 that degrees in scale-free graphs follow a power-law distribution. And as we said in the introduction, one of the principal properties of fractals concerns their scale-free complexity and the distribution of box-covers are also supposed to follow a power-law distribution (according to the definition of the Box-counting fractal dimension).

Be aware that relationship orders in these algorithms will not use V but P in $\preceq_P$ where P is a the "potential centers set" originally set to V from which we pick box centers at each iteration. P is bound to evolve (more precisely, to reduce) so $\preceq_P$ is also bound to evolve. We can have $u \prec_P v$ at a given iteration and $u \succ_P v$ at the next one.

Algorithm 7 Greedy approximation algorithm for Complete Box Cover

Require: $G = (V, E)$, $\varepsilon \in \mathbb{R}$
Ensure: $V = \bigcup_{c \in C} \mathcal{B}(c, \varepsilon)$
  $P \leftarrow V$, $C \leftarrow \emptyset$
  while $|P| \neq 0$ **do**
    Choose c in P according to CBC Heuristic
    $P \leftarrow P \setminus \mathcal{B}(c, \varepsilon)$
    $C \leftarrow C \cup \{c\}$
  end while

Algorithm 8 Greedy algorithm for Disjoint Box Cover

Require: $G = (V, E)$, $\varepsilon \in \mathbb{R}$
Ensure: $(\mathcal{B}(d, \varepsilon))_{d \in D}$ are pairwise disjoint.
  $P \leftarrow V$, $D \leftarrow \emptyset$
  while $|P| \neq 0$ **do**
    Choose d in P according to DBC Heuristic
    $P \leftarrow P \setminus \mathcal{B}(d, \varepsilon)$
    $D \leftarrow D \cup \{d\}$
    $P \leftarrow P \setminus \{p \in P \mid \mathcal{B}(d, \varepsilon) \cap \mathcal{B}(p, \varepsilon) \neq \emptyset\}$ ▷ i.e we remove from potential centers the vertices whose boxes would intersect with the box added to D
  end while

In order to prove that these algorithms are not approximations of box cover problems, we need to show examples of graphs where the greedy algorithm returns a solution as bad as wanted in comparison to the optimal solution. We will construct two graph sequences, one for each greedy algorithm.

6.1.2 *Graph sequence for CBC.*

Theorem 6.1. *Algorithm 7 is not an approximation for CBC.*

Proof. Little reminder: the star graph sequence $(S_n)_{n \geq 1}$ is defined such that S_n is the complete bipartite graph $K_{1,n}$, in other words a tree with 1 internal node and n leaves (or only 1 vertex if $n = 1$).

$\forall n \geq 3$, we construct C_n using 2 times S_n by merging the internal node a of the first one with one leaf of the second one and by merging the internal node b of the second one with one leaf of the first one. We call a_i and b_j the $n-1$ reaming leaves of the first and the second S_n. In Fig. 20, we show C_3 and C_4.

Let's apply the CBC greedy algorithm with $\varepsilon = 1$:

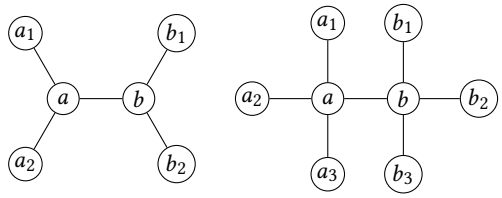

Fig. 20. C_3 and C_4 graphs

- Initialization: $G \leftarrow C_n$; $\varepsilon \leftarrow 1$; $P \leftarrow V$; $C \leftarrow \emptyset$
- First iteration: $\forall\ i, j, a =_P b \succ_P a_i =_P b_j$ $(\deg_P(b) = \deg_P(a) = n, \deg_P(a_i) = \deg_P(b_j) = 1)$; $c \leftarrow b$; (without loss of generality) $P \leftarrow \{a_i \mid i \in [\![1, n-1]\!]\}$; $C \leftarrow \{b\}$
- Next iterations: $\forall\ i,\ \deg_P(a_i) = 0$. Each $n-1$ remaining nodes of S_n need a box.
- Finalization: $P \leftarrow \emptyset$; $C \leftarrow \{b\} \cup \{a_i \mid i \in [\![1, n-1]\!]\}$
- Output: $|C| = 1 + n - 1 = n$

In simple words, we first choose one internal node as the center of the first box and suppress every node of the star graph (including the other internal node) from the potential center list. Then it remains the $n-1$ last nodes of the other star graph who will need a box for every vertex. In total, we will use n boxes. In Fig. 21, we show CBC algorithm execution on C_3 and C_4. But the optimal solution only need 2 boxes, centered on both internal nodes of S_n and S_{n+1} ($C = \{a, b\} \in \mathcal{C}(C_n, 1)$ and $|C| = 2$). Vertices chosen to be centers of boxes in CBC Greedy algorithm solution are colored in red and the additional centers of boxes in the optimal solution are colored in blue.

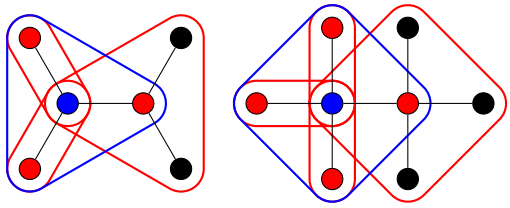

Fig. 21. CBC Algorithm on C_3 and C_4

So we can conclude that:

$$\forall n \geq 3, \frac{\text{Greedy}_{\text{CBC}}(G = C_{2n}, \varepsilon = 1)}{\text{OPT}_{\text{CBC}}(G = C_{2n}, \varepsilon = 1)} = \frac{2n}{2} = n$$

So the solution can be as bad as wanted.

Graph sequence $\forall\ \varepsilon \geq 1$. This example was only valid for $\varepsilon = 1$. To construct a solution $\forall\ \varepsilon \geq 1$, we just need to construct $C_{n,\varepsilon}$ from C_n by replacing the edge between the two internal nodes by an ε-edges path. We can easily show that the execution of the CBC greedy algorithm with max vertex heuristic and the ε given will remain the same. In Fig. 22, we show $C_{3,3}$.

□

6.1.3 *Graph sequence for DBC.*

Theorem 6.2. *Algorithm 8 is not an approximation for DBC.*

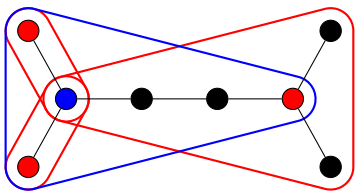

Fig. 22. $C_{3,3}$ graph

Proof. Little reminder: the complete graph sequence $(K_n)_{n\geq 1}$ is defined such that K_n is a graph with n vertices where every vertices are connected to each other.

$\forall n \geq 2$, we construct D_n using n times K_{n+2} (their vertices are called $a_{i,j}$ with $i \in [\![1, n]\!]$ and $j \in [\![1, n+2]\!]$) by connecting a vertex of each graph $(a_{i,1})$ to a "center vertex" b. In Fig. 23, we show D_2 and D_3.

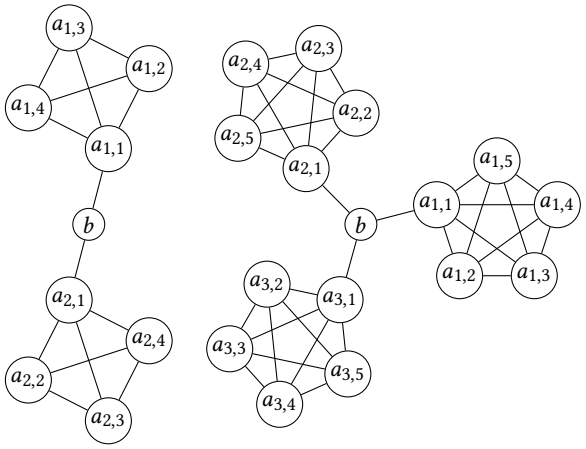

Fig. 23. D_2 and D_3 graphs

Let's apply the CBC greedy algorithm with $\varepsilon = 1$:

- Initialization: $G \leftarrow D_n$; $\varepsilon \leftarrow 1$; $P \leftarrow V$; $D \leftarrow \emptyset$
- First iteration: $\forall\ i, j,\ b \prec_P a_{i,j\neq 1} \prec_P a_{i,1}$ $(\deg_P(b) = n, \deg_P(a_{i,j\neq 1}) = n+1, \deg_P(a_{i,1}) = n+2)$; $d \leftarrow b$; $P \leftarrow \{a_{i,j} \mid (i, j) \in [\![1, n]\!] \times [\![2, n+2]\!]\}$; $D \leftarrow \{b\}$; $P \leftarrow \emptyset$
- Output: $|D| = 1$

In simple words, we first choose the center vertex and suppress every nodes of the graph from the potential center list (complete graph vertices are either directly connected to the center vertex or connected to a neighbor of the center vertex). In total, we will only have 1 box. In Fig. 24, we show DBC algorithm execution on D_2 and D_3. But optimal solution can make n boxes, by choosing a center in each complete graph (not the vertex connected to the center vertex) ($D = \bigcup_{i=1}^{n} \{a_{i,2}\} \in \mathcal{D}(D_n, 1)$ and $|D| = n$). Vertices chosen to be centers of boxes in DBC Greedy algorithm solution are colored in red and the centers of boxes in the optimal solution are colored in blue.

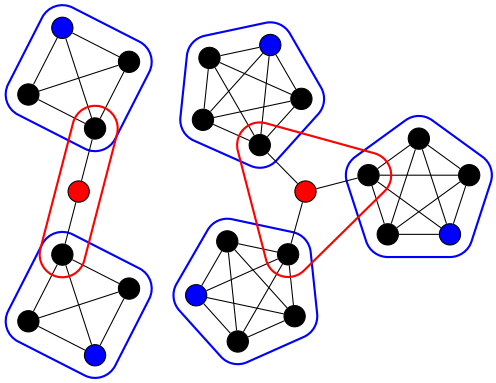

Fig. 24. DBC Algorithm on D_2 and D_3

So we conclude that:

$$\forall\, n \geq 2,\ \frac{\mathrm{OPT}_{\mathrm{DBC}}(G = D_n, \varepsilon = 1)}{\mathrm{Greedy}_{\mathrm{DBC}}(G = D_n, \varepsilon = 1)} = \frac{n}{1} = n$$

So the solution can be as bad as wanted.

Graph sequence $\forall\ \varepsilon \geq 1$. This example was only valid for $\varepsilon = 1$. To construct a solution $\forall\ \varepsilon \geq 1$, we just need to construct $D_{n,\varepsilon}$ from D_n by replacing edges between the center vertex and vertices $a_{i,1}$ by an ε-edges paths. We can easily show that the execution of the DBC greedy algorithm with min vertex heuristic and the ε given will give the same result. In Fig. 25, we show $D_{2,3}$.

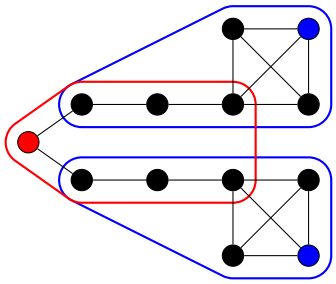

Fig. 25. $D_{2,3}$ graph

□

In conclusion, neither of the 2 greedy algorithms are approximations for their respective problem. However, we can perhaps prove that it is an approximation for a given set of fractal graphs.

6.2 Box cover conversion

These two following algorithms aim to obtain a Disjoint Box Cover from a Complete Box Cover and vice-versa such that obtained box covers sizes are as near as possible of original box cover sizes and, ideally, equal to original box covers sizes (that would mean that DCBC-DEC is true). We can note that box covers taken as arguments are not required to satisfy DBC or CBC (i.e their cardinality doesn't need to be equal to DBC or CBC). Algorithms are separated in two phases:

- 1. We choose as much centers in the argument box cover as possible thanks to BC heuristics
- 2. If the box cover obtained can possibly (or need to) be improved, we run the precedent greedy algorithms on the remaining vertices (not belonging to the original cover)

Algorithm 9 Complete Box Cover to Disjoint Box Cover provider

Require: $G = (V, E), \varepsilon \in \mathbb{R} \mid \mathrm{DBC}(G, \varepsilon) = \mathrm{CBC}(G, \varepsilon), C \in \boldsymbol{C}(G, \varepsilon)$
Ensure: $(\mathcal{B}(d, \varepsilon))_{d \in D}$ are pairwise disjoint.
  $D \leftarrow \emptyset$, $P \leftarrow V \setminus \boldsymbol{O}(C)$
  while $|P| \neq 0$ **do**
    Choose d in P according to DBC Heuristic
    Choose c in $f_{D,P}(d)$ according to DBC Heuristic
    if $\mathcal{B}(d, \varepsilon) \cap \biguplus_{d' \in D} \mathcal{B}(d', \varepsilon) = \emptyset$ **then**
      $P \leftarrow P \setminus \mathcal{B}(c, \varepsilon)$
      $D \leftarrow D \cup \{d\}$
    else
      $P \leftarrow P \setminus \{d\}$
    end if
  end while
  $V' \leftarrow V \setminus \biguplus_{d \in D} \mathcal{B}(d, \varepsilon)$
  if $V' \neq \emptyset$ **then**
    $D' \leftarrow \mathrm{AlgoDBC}\ (G[V'], \varepsilon)$
    $D \leftarrow D \cup D'$
  end if

Algorithm 10 Disjoint Box Cover to Complete Box Cover provider

Require: $G = (V, E), \varepsilon \in \mathbb{R}, D \in \mathcal{D}(G, \varepsilon)$
Ensure: $V = \bigcup_{c \in C} \mathcal{B}(c, \varepsilon)$
  $C \leftarrow \emptyset$, $P \leftarrow V \setminus \boldsymbol{\mathcal{U}}(D)$
  while $|P| \neq 0$ **do**
    Choose c in P according to CBC Heuristic
    $d \leftarrow f_{C,P}(c)$
    $P \leftarrow P \setminus \mathcal{B}(d, \varepsilon)$
    $C \leftarrow C \cup \{c\}$
  end while
  $V' \leftarrow V \setminus \bigcup_{c \in C} \mathcal{B}(c, \varepsilon)$
  if $V' \neq \emptyset$ **then**
    $C' \leftarrow \mathrm{AlgoCBC}\ (G[V'], \varepsilon)$
    $C \leftarrow C \cup C'$
  end if

6.2.1 *Algorithms.*

Optimality of conversion algorithms. When we provide as arguments under-optimal disjoint box covers (disjoint box covers too small) / over-optimal complete box covers (complete box covers too big) or graphs which doesn't satisfy DCBC-DEC (optimal disjoint box covers size doesn't match with optimal complete box covers), it's reasonable to assume that algorithms 9 and 10 will not provide good results. But what interests us is the result when graphs satisfies DCBC-DEC and box covers provided are optimal. Unfortunately, even in that case, we can find example as bad as wanted, exactly like in algorithms 8 and 7.

6.2.2 Graph sequence for CBC to DBC.

Theorem 6.3. *Algorithm 9 is not an approximation for CBC to DBC conversion with DCBC-DEC true and optimal box covers provided.*

Proof. $\forall n \geq 2$, we construct CD_n using 1 star graph S_{n+1} and $n-1$ complete graphs K_{n+2}. In the star graph, we call the internal node a and the leaves b_j ($j \in [\![1, n+1]\!]$). In each complete graph K_i ($i \in [\![1, n-1]\!]$), we choose an "internal" node called a_i and we call the others $b_{i,j}$ ($(i,j) \in [\![1, n-1]\!] \times [\![1, n+1]\!]$). Finally, we connect all the $b_{i,j}$ to the corresponding b_j. In Fig.26, we show CD_2 and CD_3.

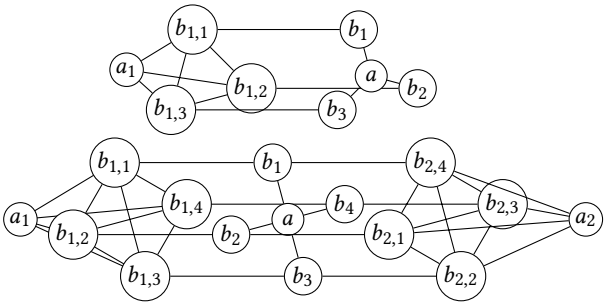

Fig. 26. CD_2 and CD_3 graphs

Let's consider $C = \{a\} \cup \{a_i \mid i \in [\![1, n-1]\!]\} \in \mathcal{C}(CD_n, 1) \cap \mathcal{D}(CD_n, 1)$ (that confirms that DCBC-DEC$(CD_n, 1)$ is true and also DBC$(CD_n, 1) = 1 + n - 1 = n$). Let's apply the CBC to DBC greedy algorithm on C with $\varepsilon = 1$:

- Initialization: $G \leftarrow CD_n$; $\varepsilon \leftarrow 1$; $P \leftarrow V$; $D \leftarrow \emptyset$
- First iteration: $\forall\ i, j,\ b_j \prec_P a =_P a_i \prec_P b_{i,j}$ ($\deg_P(b_j) = n, \deg_P(a) = \deg_P(a_i) = n+1, \deg_P(b_{i,j}) = n+2$);
  $d \leftarrow b_1$ (without loss of generality); $c \leftarrow a$;
  $P \leftarrow \{b_{i,j}, a_i \mid (i,j) \in [\![1, n-1]\!] \times [\![1, n+1]\!]\}$; $D \leftarrow \{a_1\}$;
- Next iterations: we can't add other boxes because they all would intersect with $\mathcal{B}(a_1, 1)$. So $P \leftarrow \emptyset$
- Output: $|D| = 1$

In simple words, we first choose one of the leaves of S_{n+1} as the center of the first box, suppress every node of the corresponding box of C (which is the box centered on the internal note of S_{n+1}) from the potential centers and all other vertices can't be centers because they are too close to the disjoint box chosen (including internal nodes of all the K_{n+2}). In total, we will only have 1 box. In Fig. 27, we show CBC to DBC algorithm execution on CD_2 and CD_3. But the optimal solution can make n boxes (C is an optimal solution). Vertices chosen to be centers of boxes in CBC to DBC Greedy algorithm solution are colored in red and the centers of boxes in the optimal solution are colored in blue.

So we can conclude that:

$$\forall n \geq 2, \frac{\text{OPT}_{\text{CBCtoDBC}}(G = CD_n, \varepsilon = 1)}{\text{Greedy}_{\text{CBCtoDBC}}(G = CD_n, \varepsilon = 1)} = \frac{n}{1} = n$$

So the solution can be as bad as wanted. □

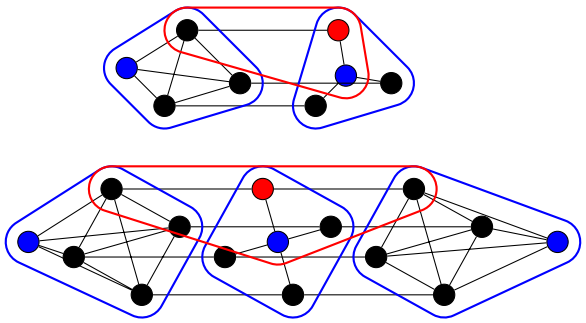

Fig. 27. CBC to DBC Algorithm on CD_2 and CD_3 graphs

6.2.3 Graph sequence for DBC to CBC.

Theorem 6.4. *Algorithm 10 is not an approximation for DBC to CBC conversion with DCBC-DEC true and optimal box covers provided.*

Proof. $\forall n \geq 3$, we construct DC_n using 2 star graph S_{n-1}. In the first star graph, we call the internal node a and the leaves a_i ($i \in [\![1, n-1]\!]$). In the other star graph, we call the internal nose b and the leaves b_j ($j \in [\![1, n-1]\!]$). Finally, we connect all the a_i to b_1. In Fig.28, we show DC_3 and DC_4.

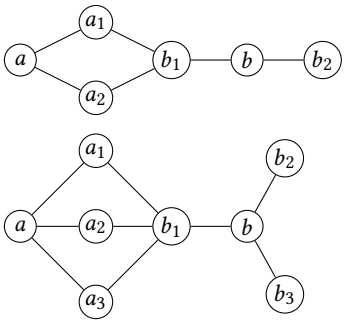

Fig. 28. DC_3 and DC_4 graphs

Let's consider $D = \{a, b\} \in \mathcal{D}(DC_n, 1) \cap \mathcal{C}(DC_n, 1)$ (that confirms that DCBC-DEC$(DC_n, 1)$ is true and CBC$(DC_n, 1) = 2$). Let's apply the DBC to CBC greedy algorithm on D with $\varepsilon = 1$:

- Initialization: $G \leftarrow DC_n$; $\varepsilon \leftarrow 1$; $P \leftarrow V$; $C \leftarrow \emptyset$
- First iteration: $\forall\ i, j,\ b_1 \succ_P a =_P a_i \succ_P b_{j \neq 1}$
  ($\deg_P(b_1) = n, \deg_P(a) = \deg_P(a_i) = n-1, \deg_P(b_{j\neq 1}) = 1$);
  $c \leftarrow b_1$; $d \leftarrow b$;
  $P \leftarrow \{a\} \cup \{a_i \mid i \in [\![1, n-1]\!]\}$; $C \leftarrow \{b_1\}$;
- Second iteration: $\forall\ i,\ a \succ_P a_i$ ($\deg_P(a) = n-1; \deg_P(a_i) = 1$);
  $c \leftarrow a$; $d \leftarrow a$; $P \leftarrow \emptyset$; $C \leftarrow \{b_1, a\}$;
- Finalization: $C \notin \mathcal{C}(DC_n, 1)$. We have to apply greedy CBC algorithm on $G[\mathcal{U}(C)] = (V = \{b_i \mid i \in [\![2, n-1]\!]\}, E = \emptyset)$.
  $C' \leftarrow \mathcal{U}(C)$ (all vertices are disconnected).
  $C \leftarrow \{a\} \cup \{b_i \mid i \in [\![2, n-1]\!]\}$;
- Output: $|C| = 1 + n - 1 = n$

In simple words, we first choose the leaf b_1 of the second star graph connected to the leaves of the first star graph as the center of

the first box and suppress every node of the corresponding box of D (which is the box centered on the internal node b) from the potential centers. Then, we choose the internal node of the first graph a and suppress every node of the corresponding box of D (which is the box centered on the same internal node a) from the potential centers. Now there is no more potential centers but C still does not cover all the vertices so we apply CBC greedy algorithm on the remaining vertices. They are all disconnect so we'll need another box for every vertex. In total, we will need n boxes. In Fig.29, we show DBC to CBC algorithm execution on DC_3 and DC_4. But the optimal solution can make 2 boxes (D is an optimal solution). Vertices chosen to be centers of boxes in CBC to DBC Greedy algorithm solution are colored in red and the additional centers of boxes that were in the optimal solution are colored in blue.

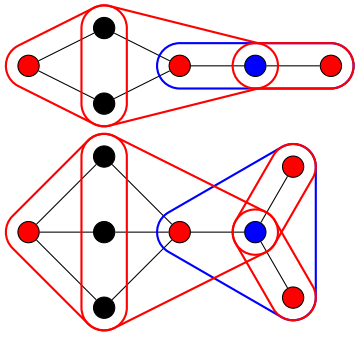

Fig. 29. DBC to CBC Algorithm on DC_3 and DC_4 graphs

So we can conclude that:

$$\forall n \geq 2, \frac{\text{Greedy}_{\text{DBCtoCBC}}(G = DC_{2n}, \varepsilon = 1)}{\text{OPT}_{\text{DBCtoCBC}}(G = DC_{2n}, \varepsilon = 1)} = \frac{2n}{2} = n$$

So the solution can be as bad as wanted. □

We haven't shown it in this paper but we can easily find graph sequences for both conversion algorithms $\forall\ \varepsilon \geq 1$. In conclusion, even if graphs satisfies DCBC-DEC and box covers provided are optimal, neither of the 2 greedy conversion algorithms are approximations for their respective problems.

7 Fractals

7.1 Formalization

Let's formalize fractal graphs. Every fractal F is the limit of its iterations F_n ($n \in \mathbb{N}$):

$$F = \lim_{n \to +\infty} F_n$$

Be aware that this fractal $F = (V_F, E_F)$ isn't a graph in the conventional sense of the term: it's an infinite graph. V_F and E_F are well-defined but infinite. At each new iteration F_{n+1}, we add new vertices and we modify edges but we keep the vertices from the precedent iteration F_n ($V_{F_n} \subseteq V_{F_{n+1}}$).

We will study fractals corresponding to sets with internal homotheties of the same ratio. In practice, in means that these fractals F are composed by N r-smaller versions of itself ($0 < r < 1$). We've got two ways to represent fractal graph iterations.

First, we can consider that each iteration is obtained by merging the previous iterations without affecting its size. It means that a ε at a n iteration proportionally corresponds to ε/r at the $n+1$ iteration. Fractal size is literally growing exponentially at each iteration. This corresponds to the first 3 iterations of Sierpinski Triangle on the top of Fig. 30. A problem with this way to represent graph fractals is that:

$$\lim_{n \to +\infty} \text{CBC}(F_n, \varepsilon) = +\infty$$

(it's also valid for DBC). So we can't define $D_{\text{box}}(F)$ because:

$$\lim_{\varepsilon \to 0^+} \lim_{n \to +\infty} \frac{\ln(\text{CBC}(F_n, \varepsilon))}{\ln(\frac{1}{\varepsilon})} = +\infty$$

Second version manages to fix this. Instead keeping previous iterations at the same size, we choose to reduce edge weights by a factor r at each iteration. It means that a ε at a n iteration proportionally corresponds to $r \times (\varepsilon/r) = \varepsilon$ at the $n+1$ iteration. Fractal size remains the same at each iteration. This corresponds to the first 3 iterations of Sierpinski Triangle on the bottom of Fig. 30.

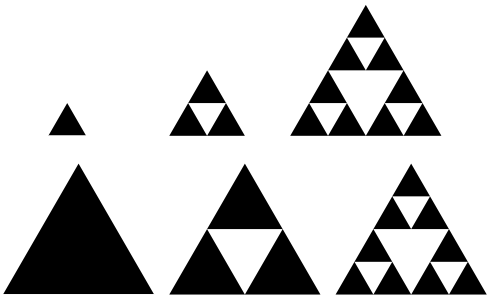

Fig. 30. The 2 different ways to deal with fractal graph scale

In the rest of this paper, F_n corresponds to the first version and $\overline{F_n}$ to the second one. To sum up, $\overline{F_n}$ corresponds to F_n with r^n weights on the edges. F_n is easier to manipulate for framing but $\overline{F_n}$ is more rigorous for box-counting dimension computation.

We limit the set of fractals with internal homotheties of the same ratio we're gonna use (that we denote $\mathcal{F}$) to fractals respecting the following property:

Proposition 7.1. $\forall\ F \in \mathcal{F},\ n \geq 1,\ u, v \in V_{F_n}$,

$$dist_{V_{\overline{F_n}}}(u, v) = dist_{V_{\overline{F_{n+1}}}}(u, v)$$

To justify why, let's focus on the Von Koch snowflake fractal V. Its ratio r is $\frac{1}{3}$. First iteration V_1 corresponds to a simple edge and we can deduce V_{n+1} from 4 different V_n by merging their respective snowflake vertices as shown in Fig. 31. At each new iteration $\overline{V_{n+1}}$, we replace each edge between 2 vertices of $V_{\overline{V_n}}$ by 4 new edges 3 times shorter. It implies that :

$$\forall\ n \geq 1,\ u, v \in V_{F_n}, \text{dist}_{V_{\overline{V_n}}}(u, v) = \frac{4}{3}\text{dist}_{V_{\overline{V_{n+1}}}}(u, v)$$

Which is a direct contradiction of Prop. 7.1. As a direct consequence:

$$\max_{u,v \in V_{\overline{V_n}}} \text{dist}(u, v) \underset{n \to +\infty}{\to} +\infty \Rightarrow \lim_{n \to +\infty} CBC(\overline{V_n}, \varepsilon) = +\infty$$

So we see that without Prop. 7.1, we can't define Box-Counting dimension with fractal graphs.

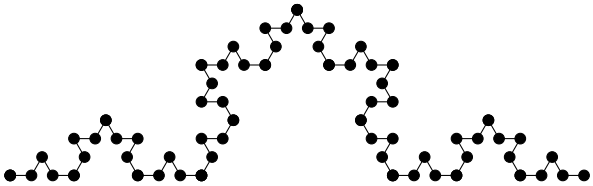

Fig. 31. V_4 Von Koch Snowflake graph

On the other hand, we've also got:

Proposition 7.2. $\forall\, F \in \mathcal{F},\ \exists\, K \in \mathbb{N} \mid \forall\, n \geq 1,$

$$\forall\, v \in V_{\overline{F_{n+1}}},\ \min_{u \in V_{\overline{F_n}}} dist(u,v) \leq K r^n$$

This property is directly deduced by induction from the construction of every $\overline{F_n}$ graph. Indeed, even fractal graphs which doesn't respect Prop. 7.1 respects Prop. 7.2. For example, for V, each new iteration V_{n+1} add vertices at most at a distance $2 \times \frac{1}{3}^n$ from the precedent iteration vertices.

Finally, let's focus on the definition of box-counting dimension. We denote $D^c_{\underline{\text{box}}}(F)$ or $D^d_{\underline{\text{box}}}(F)$ the box-counting dimensions defined with $\text{CBC}(F,\varepsilon)$ or $\text{DBC}(F,\varepsilon)$ instead of $N(\varepsilon)$ (cf Sect. 1). $D^c_{\overline{\text{box}}}(F)$, $D^d_{\overline{\text{box}}}(F)$, $D^c_{\underline{\text{box}}}(F)$, and $D^d_{\underline{\text{box}}}(F)$ are defined as the upper and lower bounds of the subsequential limit set of $D^c_{\underline{\text{box}}}(F)$ and $D^d_{\underline{\text{box}}}(F)$ if it does not converge. As a reminder, a subsequential limit of a sequence is the limit of some subsequence. All this notations are summarized in Tab. 2.

We've got uniqueness of box-counting dimension definitions:

Theorem 7.1. $\forall\, F \in \mathcal{F}$,
If $D^c_{\underline{box}}(F)$ or $D^d_{\underline{box}}(F)$ does converge, then $D^c_{\underline{box}}(F) = D^d_{\underline{box}}(F)$.
Otherwise, $D^c_{\overline{box}}(F) = D^d_{\overline{box}}(F)$ and $D^c_{\underline{box}}(F) = D^d_{\underline{box}}(F)$.

Proof. Let's consider this inequality:

$$\frac{\ln(\text{CBC}(F,2\varepsilon))}{\ln(\frac{1}{\varepsilon})} \underset{\text{Th. 2.26}}{\leq} \frac{\ln(\text{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})} \underset{\text{Cor. 2.3.1}}{\leq} \frac{\ln(\text{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

$D(F)$	$D^c(F)$	$D^d(F)$
$D_{\underline{\text{box}}}(F)$	$\lim_{\varepsilon\to 0^+} \frac{\ln(\text{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$	$\lim_{\varepsilon\to 0^+} \frac{\ln(\text{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$
$D_{\overline{\text{box}}}(F)$	$\overline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\text{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$	$\overline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\text{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$
$D_{\underline{\text{box}}}(F)$	$\underline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\text{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$	$\underline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\text{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$

Table 2. Box-counting fractal dimension notations

$$\ln\left(\frac{1}{\varepsilon}\right) = \ln\left(\frac{1}{2\varepsilon}\right) + \ln(2) = \ln\left(\frac{1}{2\varepsilon}\right)\left(1 + \frac{\ln(2)}{\ln\left(\frac{1}{2\varepsilon}\right)}\right)$$

$$1 + \frac{\ln(2)}{\ln\left(\frac{1}{2\varepsilon}\right)} \underset{\varepsilon\to 0^+}{\to} 1$$

So by passing to the limit (if it exists), thanks to the sandwich theorem, we've got equality between the limits. Otherwise, we've got the equality between the subsequential limits set upper and lower bounds. □

Since we've got uniqueness of box-counting dimension, we're only gonna use the following notations: $D_{\text{box}}(F)$, $D_{\overline{\text{box}}}(F)$ and $D_{\underline{\text{box}}}(F)$ ($D_{\underline{\text{box}}}(F) \leq D_{\overline{\text{box}}}(F)$).

Let's prove we can exchange box-counting dimension limits without a problem between F and $\overline{F_n}$.

7.2 Box-counting dimension limits

Corollary 7.1. $\forall\, F \in \mathcal{F},\ \exists\, K \in \mathbb{N} \mid \forall\, N \geq 1,\ n \geq N$

$$\forall\, v \in V_{\overline{F_n}},\ \min_{u \in V_{\overline{F_N}}} dist(u,v) \leq K \sum_{i=N}^{n-1} r^i$$

Proof. Obtained by applying a simple proof by induction to proposition Prop. 7.2. Base case is directly given by Prop. 7.2. As for the induction step, if we have the property for a given n (this is the induction hypothesis $\mathcal{H}(n)$), thanks to the triangular inequality of *dist*, $\forall\, v \in V_{\overline{F_{n+1}}}$:

$$\min_{u \in V_{\overline{F_N}}} \text{dist}(u,v)$$

$$\underset{\text{Prop. 7.1}}{\leq} \min_{(u,w) \in V_{\overline{F_N}} \times V_{\overline{F_n}}} \text{dist}(u,w) + \min_{w' \in V_{\overline{F_n}}} \text{dist}(w',v)$$

$$\underset{\mathcal{H}(n) \text{ and Prop. 7.2}}{\leq} K \sum_{i=N}^{n-1} r^i + K r^n = K \sum_{i=N}^{(n+1)-1} r^i$$

So the property is true $\forall\, n \geq N$. □

Corollary 7.2. $\forall\, F \in \mathcal{F},\ \exists\, K \in \mathbb{N} \mid \forall\, N \geq 1, n \geq N$

$$\forall\, v \in V_{\overline{F_n}},\ \min_{u \in V_{\overline{F_N}}} dist(u,v) \leq K \frac{r^N}{1-r}$$

Proof. By passing Cor. 7.1 to the limit and the strict growth of the sequence corresponding to the convergent series, we obtain $\forall\, N \geq 1,\ n \geq N$:

$$\forall\, v \in V_{\overline{F_n}},\ \min_{u \in V_{\overline{F_N}}} \text{dist}(u,v) \leq K \sum_{i=N}^{+\infty} r^i = K \frac{r^N}{1-r}$$

This convergence result is valid thanks to the d'Alambert's criterion ($|r| < 1$). □

Let's fix $F \in \mathcal{F}$ and let's denote $\forall\, \varepsilon > 0$ the two sequences $(c_n(\varepsilon))_{n\geq 1}$ and $(d_n(\varepsilon))_{n\geq 1}$ such that:

$$c_n(\varepsilon) = \text{CBC}(\overline{F_n}, \varepsilon) \quad \text{and} \quad d_n(\varepsilon) = \text{DBC}(\overline{F_n}, \varepsilon)$$

Theorem 7.2. *$(c_n(\varepsilon))_{n\geq 1}$ and $(d_n(\varepsilon))_{n\geq 1}$ are bounded.*

Proof. First, let's prove the following lemma:

Lemma 7.3. $\forall\, \varepsilon > 0, \exists\, N \geq 1 \mid \forall\, n \geq N, V_{\overline{F_N}} \in C(\overline{F_n}, \varepsilon)$

Proof. $K\frac{r^N}{1-r} \leq \varepsilon \Leftrightarrow N \leq \frac{\ln(\varepsilon(1-r))-\ln(K)}{\ln(r)}$

So we just have to choose $N = \left\lfloor \frac{\ln(\varepsilon(1-r))-\ln(K)}{\ln(r)} \right\rfloor$ in order to obtain $\forall\, n \geq N$:

$$\forall\, v \in V_{\overline{F_n}}, \min_{u \in V_{\overline{F_N}}} \operatorname{dist}(u,v) \underset{\text{Cor. 7.2}}{\leq} \varepsilon \Leftrightarrow \exists\, u \in V_{\overline{F_N}} \mid v \in \mathcal{B}(u,\varepsilon)$$

$$\Rightarrow \bigcup_{u \in V_{\overline{F_N}}} \mathcal{B}(u,\varepsilon) = V_{\overline{F_n}} \Leftrightarrow V_{\overline{F_N}} \in C(\overline{F_n}, \varepsilon) \qquad \square$$

This lemma allows us to obtain:

$$V_{\overline{F_N}} \in C(\overline{F_n}, \varepsilon) \underset{\text{Prop. 2.5}}{\Rightarrow} \operatorname{CBC}(\overline{F_n}, \varepsilon) \leq |V_{\overline{F_N}}|$$

We can therefore conclude that $\forall\, n \geq N$:

$$1 \underset{\text{Prop. 2.2}}{\leq} \operatorname{DBC}(\overline{F_n}, \varepsilon) \underset{\text{Cor. 2.3.1}}{\leq} \operatorname{CBC}(\overline{F_n}, \varepsilon) \underset{\text{Lem. 7.3}}{\leq} |V_{\overline{F_N}}| \qquad \square$$

This theorem ensures that we control $c_n(\varepsilon)$ and $d_n(\varepsilon)$ between two bounds. It also proves that $\operatorname{CBC}(F,\varepsilon)$ and $\operatorname{DBC}(F,\varepsilon)$ are well defined (and not infinite). This consideration implies that some of the complete or disjoint boxes contains an infinite quantity of vertices of F.

Theorem 7.4. $\forall\, F \in \mathcal{F}$, *depending on the convergence of* $D_{box}(F)$*:*

$$D_{box}(F) = \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln(c_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln(d_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

$$D_{\overline{box}}(F) = \overline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(c_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \overline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(d_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

$$D_{\underline{box}}(F) = \underline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(c_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \underline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(d_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

Proof. Let's denote $\underline{d(\varepsilon)}$ the lower bound of the subsequential limits sets of $(d_n(\varepsilon))_{n\geq 1}$ and $\overline{c(\varepsilon)}$ the upper bound of the subsequential limits sets of $(c_n(\varepsilon))_{n\geq 1}$. Thanks to their definition and to theorem Th. 7.2, we know that:

$$\exists\, N_d(\varepsilon) \geq 1 \mid \forall n \geq N_d(\varepsilon),\ \underline{d(\varepsilon)} \leq d_n(\varepsilon)$$

$$\exists\, N_c(\varepsilon) \geq 1 \mid \forall n \geq N_c(\varepsilon),\ \overline{c(\varepsilon)} \geq c_n(\varepsilon)$$

And on the other hand we can at most put $\underline{d(\varepsilon)}$ disjoint boxes on the graph F and we need at least $\overline{c(\varepsilon)}$ complete boxes to cover vertices of the graph F so:

$$\underline{d(\varepsilon)} \geq \operatorname{DBC}(F,\varepsilon) \quad \text{and} \quad \overline{c(\varepsilon)} \leq \operatorname{CBC}(F,\varepsilon)$$

So we can link all this inequalities into a single one $\forall\ n \geq \max(N_d(\varepsilon), N_c(\varepsilon))$:

$$\operatorname{DBC}(F,\varepsilon) \leq d_n(\varepsilon) \underset{\text{Cor. 2.3.1}}{\leq} c_n(\varepsilon) \leq \operatorname{CBC}(F,\varepsilon)$$

Thanks to the sandwich theorem and the theorem Th. 7.1, we know that if $D_{\text{box}}(F)$ limit exists, then:

$$D_{\text{box}}(F) = \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln(d_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln(c_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

Otherwise (if $D_{\text{box}}(F)$ limit does not exist):

$$D_{\overline{\text{box}}}(F) = \overline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(d_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \overline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(c_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

$$D_{\underline{\text{box}}}(F) = \underline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(d_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \underline{\lim_{\varepsilon\to 0^+}} \lim_{n\to+\infty} \frac{\ln(c_n(\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$

$\square$

7.3 Studied fractals

In this paper, we chose to study three different simple fractals in $\mathcal{F}$, with a different specification for each one. The goal is to verify how fractal features impact fractal dimension computation.

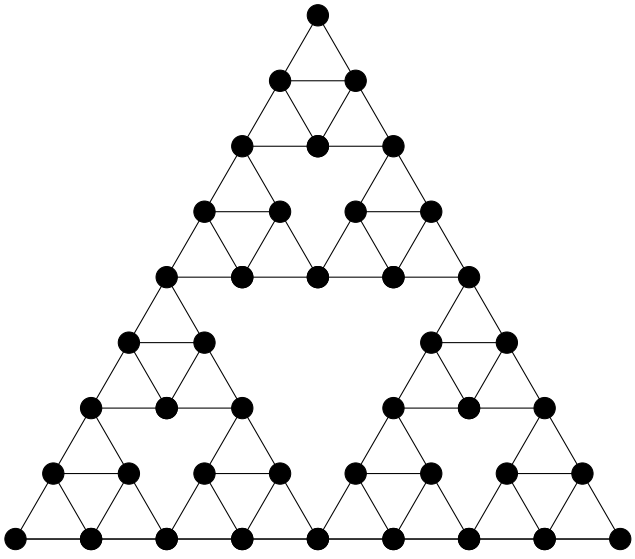

Fig. 32. S_4 Sierpinski Triangle graph

7.3.1 Sierpinski Triangle. First, we're gonna study of the simplest and best-know fractal: the Sierpinski Triangle S. First iteration S_1 corresponds to the triangle graph and we can deduce S_{n+1} from 3 different S_n by merging their respective triangle vertices as shown in Fig. 32. We can easily compute the number of vertices:

$$|V_{S_{n+1}}| = 3 \times |V_{S_n}| - 3 \Leftrightarrow |V_{S_n}| = \frac{3}{2}(3^{n-1}+1)$$

We can also easily compute the number of lines w_{S_n} (it will be useful later):

$$w_{S_{n+1}} = 2 \times w_{S_n} - 1 \Leftrightarrow w_{S_n} = 2^{n-1} + 1$$

Theorem 7.5. $D_H(S) = \frac{\ln(3)}{\ln(2)}$

Proof. By multiplying all their dimensions by 2 (the inverse of the homothety ratio), we obtain 3 new objects of the original size (the number of homotheties). Then the result comes from the direct application of the homothety dimension equivalent to the Hausdorff dimension. $\square$

Please note that we especially chose this fractal because Sierpinski Triangle nodes degrees are greater that one. It seems to us that graphical modelling of fractals would make no sense in the opposite case, since there would be no difference between a simple line and a Von Koch snowflake V, to take an example.

The K in the proposition Prop. 7.2 for S is 1.

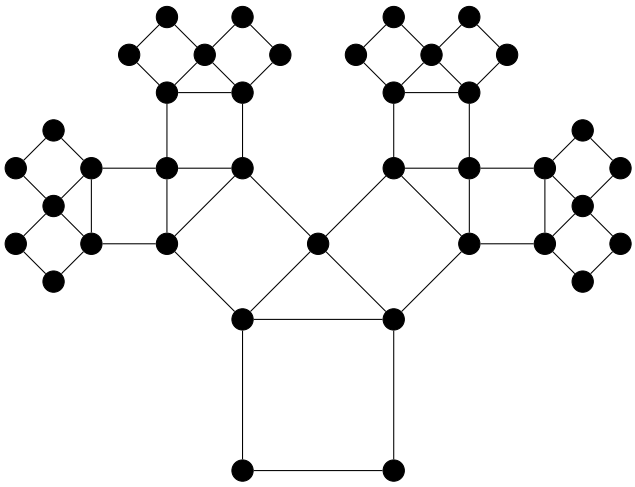

Fig. 33. P_4 Pythagoras Tree graph

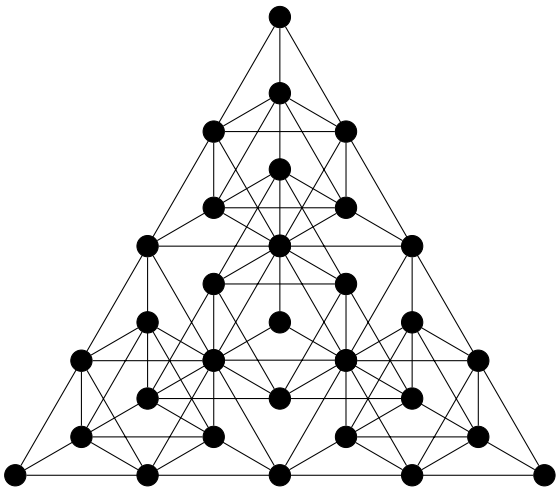

Fig. 34. T_3 Sierpinski Tetrahedron graph

7.3.2 Pythagoras Tree. Second, we're gonna study the Pythagoras Tree P. First iteration P_1 correspond to the square graph and we deduce P_{n+1} from 2 different P_n by merging one of their vertices and adding two new vertices on the basis as shown in Fig. 33.

Theorem 7.6. $D_H(P) = 2$

Proof. By applying trivial geometry, we can obtain that square side is reducing by a ratio $\sqrt{2}$ at each iteration (angle between two squares is 45°). So by multiplying all their dimension by $\sqrt{2}$ (the inverse of the homothety ratio), we obtain 2 new objects (the number of homotheties). Then, when we compute the formula of the homothety dimension equivalent to the Hausdorff dimension, we obtain: $D_{h'}(P) = \frac{\ln(2)}{\ln(\sqrt{2})} = \frac{\ln(2)}{\frac{1}{2}\ln(2)} = 2$ □

Please note that we especially chose this fractal because that in order to reproduce nodes disposition, we will give to every edge a weight proportional to the reduction of square side. It's a way for us to verify that fractal dimension computation for weighted graphs is exactly the same that for unweighted graphs.

The K in the proposition Prop. 7.2 for P is 2.

7.3.3 Sierpinski Tetrahedron. Third, we're gonna study the Sierpinski Tetrahedron T. First iteration T_1 correspond to the tetrahedron graph and we deduce T_{n+1} from 4 different T_n by merging their respective triangle vertices as shown in Fig. 34. We can notice that the number of lines w_{T_n} of T_n is equal to w_{S_n} and that the number of vertices on a face x_{T_n} of T_n is equal to $|V_{S_n}|$. Finally, we can easily compute the number of vertices:

$$|V_{T_{n+1}}| = 4 \times |V_{T_n}| - 6 \Leftrightarrow |V_{T_n}| = 2(4^{n-1} + 1)$$

Theorem 7.7. $D_H(T) = 2$

Proof. By multiplying all their dimensions by 2 (the inverse of the homothety ratio), we obtain 4 new objects of the original size (the number of homotheties). Then the result comes from the direct application of the homothety dimension equivalent to the Hausdorff dimension. □

Please note that we especially chose this fractal because it's a way for us to verify that dimensions upper that 2 are not a problem in fractal graphs computation. Indeed, manipulating fractal graphs is advantageous than manipulating fractal figures because graphs doesn't have 3-dimensional limitations.

The K in the proposition Prop. 7.2 for T is 1.

8 Framing

8.1 Tools for fractal analysis

In order to study fractals, we're going to use some new notations.

$\forall F \in \mathcal{F},\ \varepsilon \in \mathbb{R}_+^*$, we denote $\eta_c(F, \varepsilon)$ and $\eta_d(F, \varepsilon)$ the index of the last iteration whose CBC minimal is composed of one ε-box and the index of the first iteration whose DBC maximal is composed of more than one ε-box:

$$\eta_c(F, \varepsilon) = \max\{n \mid \mathrm{CBC}(F_n, \varepsilon) = 1\}$$

$$\eta_d(F, \varepsilon) = \min\{n \mid \mathrm{DBC}(F_n, \varepsilon) > 1\}$$

Theorem 8.1. $\eta_c(F, \varepsilon) = \max\{n \mid \varepsilon \geq \varepsilon_c^{n_c}(F_n)\}$

Proof. Directly derived from the equivalence:
$\varepsilon \geq \varepsilon_c^{n_c}(G) \Leftrightarrow \mathrm{CBC}(G, \varepsilon) = 1.$ □

Theorem 8.2. $\eta_d(F, \varepsilon) = \min\{n \mid \varepsilon < \varepsilon_d^{n_d}(F_n)\}$

Proof. Directly derived from the equivalence:
$\varepsilon < \varepsilon_d^{n_d}(G) \Leftrightarrow \mathrm{DBC}(G, \varepsilon) > 1$ □

In the following section, each time we'll have to compute $\eta_d(F, \varepsilon)$ and $\eta_d(F, \varepsilon)$, we'll always compute first $\varepsilon_c^{n_c}(F_n)$ and $\varepsilon_d^{n_d}(F_n)$ to make calculations easier.

8.2 Fractal box covers framing

8.2.1 Sierpinski Triangle Framing. We're going to try to limit CBC and DBC for Sierpinski Triangle by finding an upper limit for CBC and a lower limit for DBC and compare the results with what we know about them.

Upper limit for $CBC(S_n, \varepsilon)$.

THEOREM 8.3. $\forall\ n,\ \varepsilon \geq 1,\ CBC(S_n, \varepsilon) \leq \frac{3^{n-2+\ln_2(3)}}{\varepsilon^{\ln_2(3)}}$

PROOF. Thanks to the fractal construction method, if we know how to cover a given iteration with a given ε, we just have to take 3 times this cover to cover the next iteration:

$$\mathrm{CBC}(S_{n+1}, \varepsilon) \leq 3 \times \mathrm{CBC}(S_n, \varepsilon)$$

So we can define the upper limit $u(S_n, \varepsilon)$ as follows:

$$u(S_{n+1}, \varepsilon) = 3 \times u(S_n, \varepsilon) \Leftrightarrow u(S_n, \varepsilon) = 3^{n-n_0} \times u(S_{n_0}, \varepsilon)$$

But now we have to find the initialization. The idea is to consider that the upper limit equal 1 until ε is no longer sufficient to cover the entire graph with one box and then we can apply our recurrence relation. And in order to do that, we need $\eta_c(S, \varepsilon)$.

LEMMA 8.4. $\forall\ \varepsilon \geq 2,\ \eta_c(S, \varepsilon) = \lfloor \ln_2(\frac{\varepsilon}{3}) \rfloor + 3$

PROOF. Let's consider the minimum ε such that one ε-box is enough to cover the entire graph (it's $\varepsilon_c^{n_c}(S_n)$). S_n is composed of 3 sub-fractals S_{n-1}^i ($i \in [\![0, 2]\!]$). Let's name $v_{i,j}$ the common vertex between S_{n-1}^i and S_{n-1}^j ($i < j \in [\![0, 2]\!]$). We assume without loss of generality that the center vertex u of the only box covering all S_n is in S_{n-1}^0. To minimize ε, we need to minimize $\max_{v \in S_n} \mathrm{dist}(u, v)$. $\forall\ i \in [\![1, 2]\!],\ \ (v, v') \in S_{n-1}^0 \times S_{n-1}^i, \mathrm{dist}(u, v) \leq \mathrm{dist}(u, v')$ so:

$$\max_{v \in S_n} \mathrm{dist}(u, v) = \max\left(\max_{v \in S_{n-1}^1} \mathrm{dist}(u, v), \max_{v \in S_{n-1}^2} \mathrm{dist}(u, v)\right)$$

$$= \max\begin{pmatrix} \mathrm{dist}(u, v_{0,1}) + \max\limits_{v \in S_{n-1}^1} \mathrm{dist}(v_{0,1}, v) \\ \mathrm{dist}(u, v_{0,2}) + \max\limits_{v \in S_{n-1}^2} \mathrm{dist}(v_{0,2}, v) \end{pmatrix}$$

But $\forall\ v \in S_{n-1}^i, \mathrm{dist}(v_{0,i}, v) \leq w_{S_{n-1}} - 1$ (equality reached with $v = v_{1,2}$) so:

$$\max_{v \in S_n} \mathrm{dist}(u, v) = \max(\mathrm{dist}(u, v_{0,1}), \mathrm{dist}(u, v_{0,2})) + w_{S_{n-1}} - 1$$

So we have to choose u as near as possible from v_1 and v_2 simultaneously. When $n \geq 3$, it's simple, it's the vertex in the middle of the direct path between v_1 and v_2 so $\max(\mathrm{dist}(u, v1), \mathrm{dist}(u, v2)) = \frac{w_{S_{n-1}} - 1}{2} = w_{S_{n-2}} - 1$. Fig. 35 illustrates this situation. So:

$$\max_{v \in S_n} \mathrm{dist}(u, v) = w_{S_{n-2}} - 1 + w_{S_{n-1}} - 1 = 2^{n-3} + 2^{n-2} = 3 \times 2^{n-3}$$

Finally, we can conclude that $\forall\ n \geq 3,\ \varepsilon_c^{n_c}(S_n) = 3 \times 2^{n-3}$.

$$\forall\ n \geq 3,\ \varepsilon_c^{n_c}(S_n) \leq \varepsilon \Leftrightarrow 3 \times 2^{n-3} \leq \varepsilon$$
$$\Leftrightarrow n \leq \ln_2(\frac{\varepsilon}{3}) + 3 \Leftrightarrow n \leq \lfloor \ln_2(\frac{\varepsilon}{3}) \rfloor + 3$$

Using theorem Th. 8.1, we can conclude that:

$$\eta_c(S, \varepsilon) = \lfloor \ln_2(\tfrac{\varepsilon}{3}) \rfloor + 3 \qquad \square$$

Finally, we have our upper limit sequence defined $\forall\ \varepsilon \geq 2$ as follows:

$$\forall\ n \leq \eta_c(S, \varepsilon),\ u(S_n, \varepsilon) = 1$$
$$\forall\ n \geq \eta_c(S, \varepsilon),\ u(S_n, \varepsilon) = 3^{n-\eta_c(S,\varepsilon)} \times 1 = 3^{n-\lfloor \ln_2(\frac{\varepsilon}{3}) \rfloor - 3}$$

But $\lfloor \ln_2(\frac{\varepsilon}{3}) \rfloor \geq \ln_2(\frac{\varepsilon}{3}) - 1$ and $3^{-\ln_2(\varepsilon)} = \varepsilon^{-\ln_2(3)}$ so:

$$u(S_n, \varepsilon) \leq 3^{n-2-\ln_2(\frac{\varepsilon}{3})} = \frac{3^{n-2+\ln_2(3)}}{\varepsilon^{\ln_2(3)}} = \overline{u}(S_n, \varepsilon) \qquad \square$$

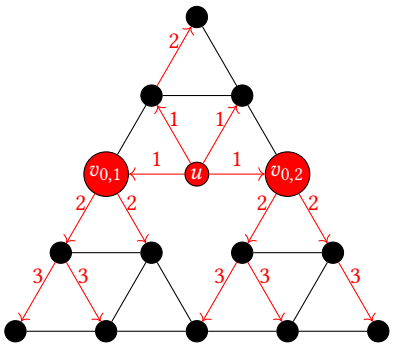

Fig. 35. S_3 covered by $\mathcal{B}(u, 3 \times 2^{3-3} = 3)$

Lower limit for $DBC(S_n, \varepsilon)$.

THEOREM 8.5. $\forall\ n,\ \varepsilon \geq 1,\ DBC(S_n, \varepsilon) \geq \frac{3^{n-2}}{2\varepsilon^{\ln_2(3)}}$

PROOF. Thanks to the fractal construction method, if we know how to cover a given iteration with a given ε, we just have to take 3 times this cover to cover the next iteration. But there can be some overlaps so we need to take back 3 boxes (overlaps can happen at each common vertices between the 3 sub-fractals):

$$\mathrm{DBC}(S_{n+1}, \varepsilon) \geq 3 \times \mathrm{DBC}(S_n, \varepsilon) - 3$$

So we can define the lower limit $l(S_n, \varepsilon)$ as follows:

$$l(S_{n+1}, \varepsilon) = 3 \times l(S_n, \varepsilon) - 3 \Leftrightarrow l(S_n, \varepsilon) = 3^{n-n_0} \times (l(S_{n_0}, \varepsilon) - \frac{3}{2}) + \frac{3}{2}$$

But now we have to find the initialization. The idea is to consider that the lower limit equal 1 until we can put at least ε disjoint boxes on the graph and then we can apply our recurrence relation with the next iteration (the slight difference with the upper limit is that you don't start with a box cover whose size is 1, otherwise the sequence would stagnate). And in order to do that, we need $\eta_d(S, \varepsilon)$.

LEMMA 8.6. $\forall\ \varepsilon \geq 2,\ \eta_d(S, \varepsilon) = \lceil \ln_2(\varepsilon) \rceil + 2$

PROOF. Let's find $\varepsilon_d^{n_d}(S_n)$ by maximazing ε such that two disjoint ε-boxes can be put on the graph. We assume without loss of generality that their center vertex u_0 and u_1 are in S_{n-1}^0 and S_{n-1}^1. To maximize ε, we have to maximize $\mathrm{dist}(u_0, u_1)$.

$$\max_{(u_0,u_1) \in S_{n-1}^0 \times S_{n-1}^1} \mathrm{dist}(u_0, u_1) =$$

$$\begin{cases} \max\limits_{u_0 \in S_{n-1}^0} \mathrm{dist}(u_0, v_{0,1}) + \max\limits_{u_1 \in S_{n-1}^1} \mathrm{dist}(v_{0,1}, u_1) & \text{or} \\ \max\limits_{u_0 \in S_{n-1}^0} \mathrm{dist}(u_0, v_{0,2}) + \mathrm{dist}(v_{0,2}, v_{1,2}) + \max\limits_{u_1 \in S_{n-1}^1} \mathrm{dist}(v_{1,2}, u_1) \end{cases}$$

In other words, shortest path between u_0 and u_1 goes through $v_{0,1}$ or through $v_{0,2}$ and $v_{1,2}$. We can prove that if shortest past between u_0 and u_1 goes through $v_{0,2}$ and $v_{1,2}$ (and there's no path as short as this through $v_{0,1}$), it means that they are not as far apart as possible so:

$$\max_{(u_0,u_1) \in S_{n-1}^0 \times S_{n-1}^1} \mathrm{dist}(u_0, u_1) =$$

$$\max_{u_0 \in S_{n-1}^0} \mathrm{dist}(u_0, v_{0,1}) + \max_{u_1 \in S_{n-1}^1} \mathrm{dist}(v_{0,1}, u_1)$$

Maximum is reached with u_0 as the topmost vertex of S^0_{n-1} and u_1 as the leftmost vertex of S^1_{n-1} so ε covers up to half of the path between these vertices. We can notice that we can put one additional disjoint box on the latest rightmost vertex is S^2_{n-1} so the initialization of lower limit sequence will start with 3 disjoint boxes (and not 2). Fig. 36 illustrates this situation.

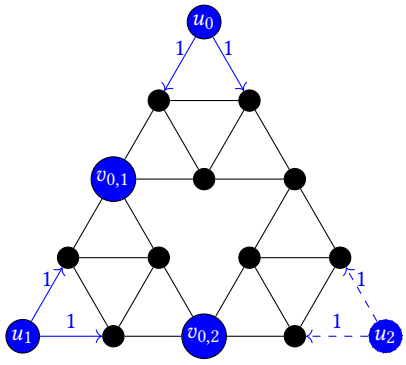

Fig. 36. S_3 containing $\mathcal{B}(u_0, 1 < 2^{3-2} = 2)$, $\mathcal{B}(u_1, 1)$ (and $\mathcal{B}(u_2, 1)$)

So finally, we can conclude that $\forall\ n \geq 3,\ \varepsilon_d^{n_d} = w_{S_{n-1}} - 1 = 2^{n-2}$.

$$\forall\ n \geq 3,\ \varepsilon_d^{n_d}(S_n) > \varepsilon \Leftrightarrow 2^{n-2} > \Leftrightarrow n > \ln_2(\varepsilon) + 2$$
$$\Leftrightarrow n \geq \lceil \ln_2(\varepsilon) + 2 \rceil \Leftrightarrow n \geq \lceil \ln_2(\varepsilon) \rceil + 2$$

Using Th. 8.2, we can conclude that:

$\forall\ \varepsilon \geq 0,\ \eta_d(S, \varepsilon) = \lceil \ln_2(\varepsilon) \rceil + 2$ □

Finally, we have our lower limit sequence defined $\forall\ \varepsilon \geq 1$ as follows:

$$\forall\ n \leq \eta_d(S, \varepsilon),\ l(S_n, \varepsilon) = 1,\ \forall n > \eta_d(S, \varepsilon),$$
$$l(S_n, \varepsilon) = 3^{n-\eta_d(S,\varepsilon)} \times (3 - \frac{3}{2}) + \frac{3}{2} = \frac{3}{2}(3^{n-\eta_d(S,\varepsilon)} + 1)$$

But $\lceil \ln_2(\varepsilon) \rceil \leq \ln_2(\varepsilon) + 1$ and $3^{-\ln_2(\varepsilon)} = \varepsilon^{-\ln_2(3)}$ so:

$$l(S_n, \varepsilon) = \tfrac{3}{2}(3^{n-\lceil \ln_2(\varepsilon) \rceil - 2} + 1) \geq \tfrac{3}{2}(3^{n-3-\ln_2(\varepsilon)} + 1)$$
$$l(S_n, \varepsilon) \geq \tfrac{3}{2} \times \frac{3^{n-3}}{\varepsilon^{\ln_2(3)}} = \frac{3^{n-2}}{2\varepsilon^{\ln_2(3)}} = \bar{l}(S_n, \varepsilon)$$ □

Framing for $CBC(S_n, \varepsilon)$ and $DBC(S_n, \varepsilon)$.

Corollary 8.6.1. $\forall\ n,\ \varepsilon \geq 1$,

$$\frac{3^{n-2}}{2\varepsilon^{\ln_2(3)}} \leq DBC(S_n, \varepsilon) \leq CBC(S_n, \varepsilon) \leq \frac{3^{n-2+\ln_2(3)}}{\varepsilon^{\ln_2(3)}}$$

Proof. Directly derived from Th. 8.3, Th. 8.5 and Cor. 2.3.1. □

This limits are exponential in n and inversely polynomial in ε. Given the fact we know CBC-DEC$(S_n, \varepsilon, |V_{S_n}|)$ is always true, we can verify that the upper limit gave us a polynomial order in ε:

$$\frac{|V_{S_n}|}{\overline{u}(S_n, \varepsilon)} = \frac{3(3^{n-1} + 1)\varepsilon^{\ln_2(3)}}{2 \times 3^{n-2+\ln_2(3)}} \underset{n \to +\infty}{\sim} \frac{3^n \times \varepsilon^{\ln_2(3)}}{2 \times 3^{n-2+\ln_2(3)}} = \frac{9\varepsilon^{\ln_2(3)}}{2 \times 3^{\ln_2(3)}}$$

More importantly, we can verify the order of our framing:

Corollary 8.6.2. *$DBC(S_n, \varepsilon)$ and $CBC(S_n, \varepsilon)$ are $\theta\left(\frac{3^n}{\varepsilon^{\ln_2(3)}}\right)$*

Proof. $\frac{\overline{u}(S_n,\varepsilon)}{\bar{l}(S_n,\varepsilon)} = \frac{\frac{3^{n-2+\ln_2(3)}}{\varepsilon^{\ln_2(3)}}}{\frac{3^{n-2}}{2\varepsilon^{\ln_2(3)}}} = 2 \times 3^{\ln_2(3)}$

That means $\overline{u}$ and $\bar{l}$ have a constant ratio. On the other side:

$$\overline{u}(S_n, \varepsilon),\ \bar{l}(S_n, \varepsilon) = \theta\left(\frac{3^n}{\varepsilon^{\ln_2(3)}}\right)$$

So thanks to these considerations, Cor. 8.6.1 inequality directly implies the corollary. This is a very good result, as we managed to have very precise control over DBC and CBC. □

Here are other framing results for Pythagoras Tree and Sierpinski Tetrahedron box covers framing. We will not be as exhaustive as for Sierpinski Triangle box cover framing.

8.2.2 *Pythagoras Tree Framing.*

Upper limit for $CBC(P_n, \varepsilon)$.

$$\mathrm{CBC}(P_{n+1}, \varepsilon) \leq 2 \times \mathrm{CBC}(P_n, \varepsilon) + 2$$
$$u(P_{n+1}, \varepsilon) = 2 \times u(P_n, \varepsilon) + 2 \Leftrightarrow u(P_n, \varepsilon) = 2^{n-n_0} \times (u(S_{n_0}, \varepsilon) + 2) - 2$$
$$\varepsilon_c^{n_c}(P_n) = \textstyle\sum_{i=0}^{n-2} \sqrt{2}^i + \sqrt{2}^{n-2} = (\sqrt{2})^{n-2}(3 + \sqrt{2}) - (1 + \sqrt{2})$$
$$\eta_c(P, \varepsilon) = \left\lfloor 2 \ln_2\left(\tfrac{\varepsilon+1+\sqrt{2}}{3+\sqrt{2}}\right)\right\rfloor + 2$$
$$u(P_n, \varepsilon) = 2^{n-\left\lfloor 2\ln_2\left(\frac{\varepsilon+1+\sqrt{2}}{3+\sqrt{2}}\right)\right\rfloor - 2} \times (1 + 2) - 2$$
$$\overline{u}(P_n, \varepsilon) = 2^{n-1} \times 3\left(\tfrac{3+\sqrt{2}}{\varepsilon+1+\sqrt{2}}\right)^2$$

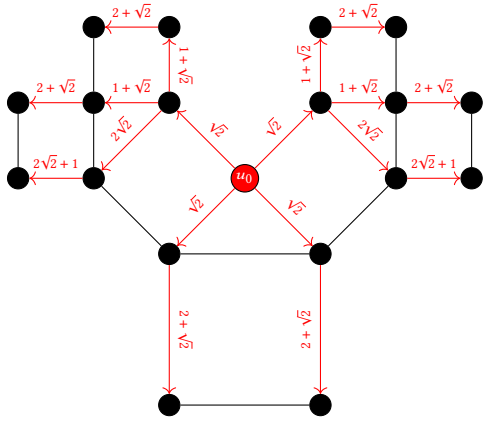

Fig. 37. P_3 covered by $\mathcal{B}(u_1, 1 + \sqrt{2} < \sum_{i=0}^{3-2} \sqrt{2}^i + \sqrt{2}^{3-2} = 1 + 2\sqrt{2})$

Theorem 8.7. $\forall\ n,\ \varepsilon \geq 1,\ CBC(P_n, \varepsilon) \leq 2^{n-1} \times 3\left(\frac{3+\sqrt{2}}{\varepsilon+1+\sqrt{2}}\right)^2$

Lower limit for $DBC(P_n, \varepsilon)$.

$$\mathrm{DBC}(P_{n+1}, \varepsilon) \geq 2 \times \mathrm{DBC}(P_n, \varepsilon) - 1$$
$$l(P_{n+1}, \varepsilon) = 2 \times l(P_n, \varepsilon) - 1 \Leftrightarrow l(P_n, \varepsilon) = 2^{n-n_0} \times (l(P_{n_0}, \varepsilon) - 1) + 1$$
$$\varepsilon_d^{n_d}(P_n) = \textstyle\sum_{i=0}^{n-2} \sqrt{2}^i + \sqrt{2}^{n-2} = (\sqrt{2})^{n-2}(3 + \sqrt{2}) - (1 + \sqrt{2})$$
$$\eta_d(P, \varepsilon) = \left\lceil 2 \ln_2\left(\tfrac{\varepsilon+1+\sqrt{2}}{3+\sqrt{2}}\right)\right\rceil + 2$$
$$l(P_n, \varepsilon) = 2^{n-\left\lceil 2\ln_2\left(\frac{\varepsilon+1+\sqrt{2}}{3+\sqrt{2}}\right)\right\rceil - 2} \times (2 - 1) + 1$$

$$\bar{l}(P_n,\varepsilon) = 2^{n-3}\left(\frac{3+\sqrt{2}}{\varepsilon+1+\sqrt{2}}\right)^2$$

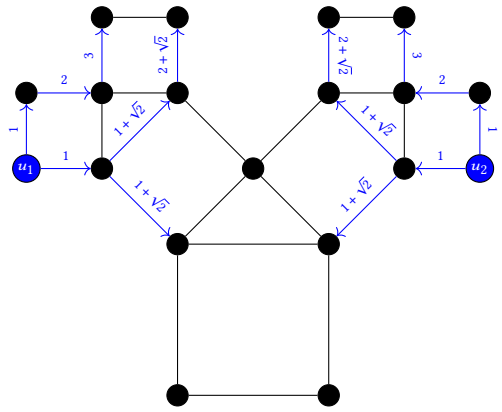

Fig. 38. P_3 containing $\mathcal{B}(u_1, 1+\sqrt{2} < \sum_{i=0}^{3-2}\sqrt{2}^i + \sqrt{2}^{3-2} = 1+2\sqrt{2})$ and $\mathcal{B}(u_1, 1+\sqrt{2})$

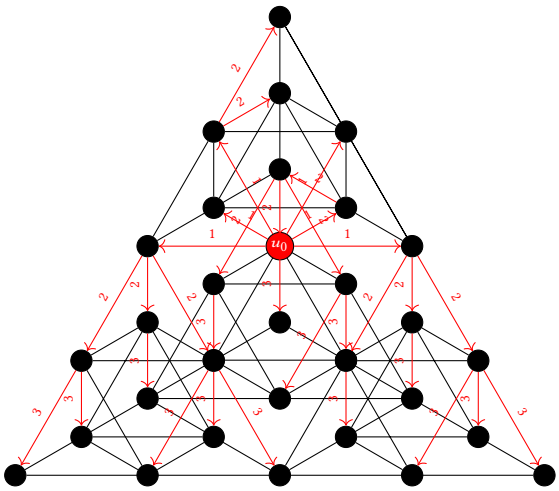

Fig. 39. T_3 covered by $\mathcal{B}(u_0, 3)$

Theorem 8.8. $\forall\, n,\ \varepsilon \geq 1,\ DBC(P_n,\varepsilon) \geq 2^{n-3}\left(\frac{3+\sqrt{2}}{\varepsilon+1+\sqrt{2}}\right)^2$

Corollary 8.8.1. $\forall\, n,\ \varepsilon \geq 1,$

$$2^{n-3}\left(\frac{3+\sqrt{2}}{\varepsilon+1+\sqrt{2}}\right)^2 \leq DBC(P_n,\varepsilon) \leq CBC(P_n,\varepsilon) \leq 2^{n-1}\times 3\left(\frac{3+\sqrt{2}}{\varepsilon+1+\sqrt{2}}\right)^2$$

$$\frac{\overline{u}(P_n,\varepsilon)}{\bar{l}(P_n,\varepsilon)} = 12 \quad \text{and} \quad \overline{u}(S_n,\varepsilon),\ \bar{l}(S_n,\varepsilon) = \theta\left(\frac{2^n}{(\varepsilon+1+\sqrt{2})^2}\right)$$

Corollary 8.8.2. *$DBC(P_n,\varepsilon)$ and $CBC(P_n,\varepsilon)$ are* $\theta\left(\frac{2^n}{(\varepsilon+1+\sqrt{2})^2}\right)$

8.2.3 *Sierpinski Tetrahedron Framing.*

Upper limit for $CBC(T_n,\varepsilon)$.

$$\mathrm{CBC}(T_{n+1},\varepsilon) \leq 4\times \mathrm{CBC}(T_n,\varepsilon)$$
$$u(T_{n+1},\varepsilon) = 4\times u(T_n,\varepsilon) \Leftrightarrow u(T_n,\varepsilon) = 4^{n-n_0}\times u(T_{n_0},\varepsilon)$$
$$\varepsilon_c^{n_c}(T_n) = w_{T_{n-3}} - 1 + w_{T_{n-2}} - 1 + w_{T_{n-1}} - 1 = 7\times 2^{n-4}$$
$$\eta_c(T,\varepsilon) = \left\lfloor \ln_2\left(\frac{\varepsilon}{7}\right)\right\rfloor + 4$$
$$u(T_n,\varepsilon) = 4^{n-\lfloor \ln_2(\frac{\varepsilon}{7})\rfloor - 4}\times 1$$
$$\overline{u}(T_n,\varepsilon) = 49\frac{4^{n-3}}{\varepsilon^2}$$

Theorem 8.9. $\forall\, n,\ \varepsilon \geq 1,\ CBC(T_n,\varepsilon) \leq 49\frac{4^{n-3}}{\varepsilon^2}$

Lower limit for $DBC(T_n,\varepsilon)$.

$$\mathrm{DBC}(T_{n+1},\varepsilon) \geq 4\times \mathrm{DBC}(T_n,\varepsilon) - 6$$
$$l(T_{n+1},\varepsilon) = 4\times l(T_n,\varepsilon) - 6 \Leftrightarrow l(T_n,\varepsilon) = 4^{n-n_0}\times(l(T_{n_0},\varepsilon) - 2) + 2$$
$$\varepsilon_d^{n_d}(T_n) = 2^{n-2}$$
$$\eta_d(T,\varepsilon) = \lceil \ln_2(\varepsilon)\rceil + 2$$
$$l(T_n,\varepsilon) = 4^{n-\lceil \ln_2(\varepsilon)\rceil - 2}\times(4-2) + 2$$
$$\bar{l}(T_n,\varepsilon) = 2\frac{4^{n-3}}{\varepsilon^2}$$

Theorem 8.10. $\forall\, n,\ \varepsilon \geq 1,\ DBC(T_n,\varepsilon) \geq 2\frac{4^{n-3}}{\varepsilon^2}$

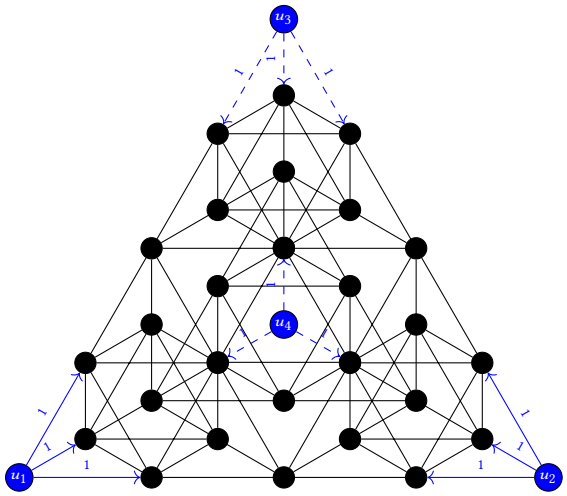

Fig. 40. T_3 containing $\mathcal{B}(u_1, 1+\sqrt{2} < 2^{3-2} = 2)$ and $\mathcal{B}(u_2, 1)$ (and $\mathcal{B}(u_3, 1)$ and $\mathcal{B}(u_4, 1)$)

Corollary 8.10.1. $\forall\, n,\ \varepsilon \geq 1,$

$$2\frac{4^{n-3}}{\varepsilon^2} \leq DBC(T_n,\varepsilon) \leq CBC(T_n,\varepsilon) \leq 49\frac{4^{n-3}}{\varepsilon^2}$$

$$\frac{\overline{u}(P_n,\varepsilon)}{\bar{l}(P_n,\varepsilon)} = \frac{49}{2} \quad \text{and} \quad \overline{u}(S_n,\varepsilon),\ \bar{l}(S_n,\varepsilon) = \theta\left(\frac{4^n}{\varepsilon^2}\right)$$

Corollary 8.10.2. *$DBC(S_n,\varepsilon)$ and $CBC(S_n,\varepsilon)$ are* $\theta\left(\frac{4^n}{\varepsilon^2}\right)$

8.3 Framing procedure

As a conclusion, we developed a framing procedure to get framing on CBC and DBC on fractal graphs on condition of sufficiently precise self-similarity conditions. To obtain upper / lower limit sequence, you need to do:

1. For a given epsilon, find an order relationship between CBC / DBC of a given fractal iteration and the next one (thanks to fractal construction method).
2. Replace the inequality by an equality and CBC / DBC by your upper / lower limit. You obtain the recurrence relationship which defines you limit sequence.
3. Compute $\varepsilon_c^{n_c}(F_n)$ / $\varepsilon_d^{n_d}(F_n)$ and then $\eta_c(F,\varepsilon)$ / $\eta_d(F,\varepsilon)$.
4. Initialize your upper / lower limit sequence with this data.
5. Simplify your upper / lower limit sequence in order to obtain easy to handle bounds for the function.

If we denote $f(F_n,\varepsilon)$ the framing function we obtained at the end of the framing procedure and N is the number of homotheties of a given fractal F, we can notice interesting links as showed in Tab. 3.

We can therefore make the following conjecture on sets with internal homotheties of the same ratio (necessary in order to ensure the existence of the number of homotheties N):

Theorem 8.11. *$\forall F \in \mathcal{F},\ \exists\, c \geq 0 \mid$ DBC(F_n,ε) and CBC(F_n,ε) are $\theta\left(\frac{N^n}{(\varepsilon+c)^{D_H(F)}}\right) \Leftrightarrow f(F_n,\varepsilon) = \frac{N^n}{(\varepsilon+c)^{D_H(F)}}$*

Proof. This proof will not be perfectly rigorous (especially for step 1 or step 3), it is rather a "proof by arguments". Let's try to apply our framing procedure to any fractal graph $F \in \mathcal{F}$.

1. Sets with internal homotheties of the same ratio ensure the possibility to extract from the fractal construction an order relationship similar to one of theses inequalities:
$$\mathrm{CBC}(F_{n+1},\varepsilon) \leq\ N_F \times \mathrm{CBC}(F_n,\varepsilon) + a$$
$$\mathrm{DBC}(F_{n+1},\varepsilon) \geq\ N_F \times \mathrm{DBC}(F_n,\varepsilon) - a$$
with $a \geq 0$ and $N_F > 1$.
2. We extract one of the following recurrence relationship:
$$u(F_{n+1},\varepsilon) = N_F \times u(F_n,\varepsilon) + a \Leftrightarrow$$
$$u(F_n,\varepsilon) = N_F^{n-n_0} \times \left(u\left(F_{n_0},\varepsilon\right) + a'\right) - a'$$
$$l(F_{n+1},\varepsilon) = N_F \times l(F_n,\varepsilon) - a \Leftrightarrow$$
$$l(F_n,\varepsilon) = N_F^{n-n_0} \times \left(l\left(F_{n_0},\varepsilon\right) - a'\right) + a'$$
with $a' \geq 0$. This formulation is possible because $N_F \neq 1$.
3. If we return to the definition of sets with internal homotheties of the same ratio, we know that in each iteration, the fractal grows by a factor equal to the inverse of the homothety ratio. That means that if we can cover a fractal iteration F_n with k ε–boxes, we'll need approximately $\frac{k}{r}$ ε–boxes to cover F_{n+1}. This implies the existence of b, c, b' and c' such that:
$$\varepsilon_c^{n_c}(F_n) = b\left(\frac{1}{r}\right)^n - c \Rightarrow \eta_c(F,\varepsilon) = \left\lfloor \ln_{\frac{1}{r}}\left(\frac{\varepsilon+c}{b}\right)\right\rfloor$$

Fractal	Sierpinski Triangle	Pythagoras Tree	Sierpinski Tetrahedron
$f(F_n,\varepsilon)$	$3^n/\varepsilon^{\ln_2(3)}$	$2^n/(\varepsilon+1+\sqrt{2})^2$	$4^n/\varepsilon^2$
N	3	2	4
$D_H(F)$	$\ln_2(3)$	2	2

Table 3. Links between fractal dimension features and framing function

$$\varepsilon_d^{n_d}(F_n) = b'\left(\frac{1}{r}\right)^n - c' \Rightarrow \eta_d(F,\varepsilon) = \left\lceil \ln_{\frac{1}{r}}\left(\frac{\varepsilon+c'}{b'}\right)\right\rceil$$
$$\forall\, n \geq 0,\ \varepsilon_c^{n_c}(F_n) \underset{\text{Cor. 2.3.1}}{\geq} \varepsilon_d^{n_d}(F_n) \Rightarrow b \geq b'$$

4. With the sequence initialization, the limit sequence will look like:
$$u(F_n,\varepsilon) = N_F^{n-\left\lfloor \ln_{\frac{1}{r}}\left(\frac{\varepsilon+c}{b}\right)\right\rfloor} \times \left(u\left(F_{\eta_c(F,\varepsilon)},\varepsilon\right) + a'\right) - a'$$
$$\leq K_u \frac{N_F^n}{(\varepsilon+c)^{\ln_{\frac{1}{r}}(N_F)}} = K_u \frac{N_F^n}{(\varepsilon+c)^{D_H(F)}} = \overline{u}(F_n,\varepsilon)$$
$$l(F_n,\varepsilon) = N_F^{n-\left\lceil \ln_{\frac{1}{r}}\left(\frac{\varepsilon+c'}{b'}\right)\right\rceil} \times \left(l\left(F_{\eta_d(F,\varepsilon)},\varepsilon\right) - a'\right) + a'$$
$$\geq K_l \frac{N_F^n}{(\varepsilon+c')^{\ln_{\frac{1}{r}}(N_F)}} = K_l \frac{N_F^n}{(\varepsilon+c')^{D_H(F)}} = \underline{l}(F_n,\varepsilon)$$
Where K_u and K_l are defined as:
$$K_u = N_F^{\ln_{\frac{1}{r}}(b)+1}(1+a') \geq 0 \quad K_l = N_F^{\ln_{\frac{1}{r}}(b')-1}(N_F - a') \geq 0$$
Because $u(F_{\eta_c(F,\varepsilon)},\varepsilon) = 1$ and $l(F_{\eta_d(F,\varepsilon)},\varepsilon) = N_F$. Therefore, we have this inequality:
$$K_l \frac{N_F^n}{(\varepsilon+c')^{D_H(F)}} \leq \mathrm{DBC}(F_n,\varepsilon) \leq \mathrm{CBC}(F_n,\varepsilon) \leq K_u \frac{N_F^n}{(\varepsilon+c)^{D_H(F)}}$$
We can deduce the following informations about c and c':
$$\forall\, \varepsilon > 0,\ K_l,\ K_u,\ \mathrm{DBC}(F_n,\varepsilon),\ \mathrm{CBC}(F_n,\varepsilon) \geq 0 \Rightarrow c',\ c \geq 0$$
$$\frac{K_u}{K_l} = N_F^{2-\ln_{\frac{1}{r}}\left(\frac{b}{b'}\right)} \frac{1+a'}{N_F - a'} \geq \frac{N_F^2}{N_F - a'} \geq N_F > 1$$
$$\Rightarrow K_l < K_u \Rightarrow c' \leq c \Rightarrow \frac{1}{(\varepsilon+c)^{D_H(F)}} \leq \frac{1}{(\varepsilon+c')^{D_H(F)}}$$
$$\Rightarrow \mathrm{DBC}(F_n,\varepsilon) \text{ and } \mathrm{CBC}(F_n,\varepsilon) \text{ are } \theta\left(\frac{N^n}{(\varepsilon+c)^{D_H(F)}}\right)$$
□

Corollary 8.11.1. *$\forall F \in \mathcal{F},\ \exists\, c \geq 0 \mid$ DBC$(\overline{F_n},\varepsilon)$ and CBC$(\overline{F_n},\varepsilon)$ are $\theta\left((\varepsilon+cr^n)^{-D_H(F)}\right) \Leftrightarrow f\left(\overline{F_n},\varepsilon\right) = (\varepsilon+cr^n)^{-D_H(F)}$*

Proof. Let's compute $f\left(\overline{F_n},\varepsilon\right)$. Since ε-boxes in $\overline{F_n}$ corresponds to $\frac{\varepsilon}{r^n}$-boxes in F_n:
$$\mathrm{CBC}\left(\overline{F_n},\varepsilon\right) = \mathrm{CBC}\left(F_n,\frac{\varepsilon}{r^n}\right) \Rightarrow f\left(\overline{F_n},\varepsilon\right) = f\left(F_n,\frac{\varepsilon}{r^n}\right)$$
So we obtain:
$$f\left(F_n,\frac{\varepsilon}{r^n}\right) \underset{\text{Th. 8.11}}{=} \frac{N^n}{\left(\frac{\varepsilon}{r^n}+c\right)^{D_H(F)}} = \frac{N^n}{(\varepsilon+cr^n)^{D_H(F)}\left(\left(\frac{1}{r}\right)^{\ln_{\frac{1}{r}}(N)}\right)^n}$$
$$\Leftrightarrow f\left(\overline{F_n},\varepsilon\right) = \frac{N^n}{(\varepsilon+cr^n)^{D_H(F)}\times N^n} = (\varepsilon+cr^n)^{-D_H(F)}$$ □

And this leads us to the most important result of this paper:

Corollary 8.11.2. *If we take $f\left(\overline{F_n},\varepsilon\right)$ as an approximation for box-counting optimum values, Schroeder's conjecture ("in case of exact self-similarity, D_H and D_{box} are equal") is true for fractals in $\mathcal{F}$.*

Proof.

$$D_{\text{box}}(F) = D_{\text{box}}\left(\lim_{n\to+\infty} \overline{F_n}\right) \underset{\text{Th. 7.4}}{=} \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln\left(\text{CBC}(F_n,\varepsilon)\right)}{\ln\left(\frac{1}{\varepsilon}\right)}$$

If we accept to take $f\left(\overline{F_n},\varepsilon\right)$ as an approximation for $\text{DBC}(\overline{F_n},\varepsilon)$ or $\text{CBC}(\overline{F_n},\varepsilon)$:

$$D_{\text{box}}(F) = \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln\left(f\left(\overline{F_n},\varepsilon\right)\right)}{\ln\left(\frac{1}{\varepsilon}\right)}$$

$$\underset{\text{Cor. 8.11.1}}{=} \lim_{\varepsilon\to 0^+} \lim_{n\to+\infty} \frac{\ln\left((\varepsilon + cr^n)^{-D_H(F)}\right)}{\ln\left(\frac{1}{\varepsilon}\right)} = \lim_{\varepsilon\to 0^+} \frac{\ln\left(\varepsilon^{-D_H(F)}\right)}{\ln\left(\frac{1}{\varepsilon}\right)}$$

$$= \lim_{\varepsilon\to 0^+} \frac{D_H(F)\ln\left(\frac{1}{\varepsilon}\right)}{\ln\left(\frac{1}{\varepsilon}\right)} = \lim_{\varepsilon\to 0^+} D_H(F) = D_H(F)$$

The weaker version of this corollary (but which doesn't require the validity of the approximation) is:

$$\forall\, F\in\mathcal{F},\ K_l \times D_H(F) \le D_{\text{box}}(F) \le K_u \times D_H(F)$$

Or, if we don't know the convergence of $D_{\text{box}}(F)$:
$\forall\, F\in\mathcal{F},\ K_l \times D_H(F) \le D_{\underline{\text{box}}}(F) \le D_{\overline{\text{box}}}(F) \le K_u \times D_H(F)$ □

Theorem Th. 8.11 and corollary Cor 8.11.2 are limited to sets with internal homotheties of the same ratio. This is quite a restriction. Could we generalize this results to sets with internal homotheties?

8.4 Generalization

Before answering this question, we first need to better understand what really are fractals with internals homotheties with different ratios. In order to explain, that, we show in Fig. 41 an iteration of the Asymmetrical Pythagoras Tree fractal graph $P_{n,\alpha}$. The difference with Pythagoras Tree fractal graph P_n lies in the α angle between the basis square and the next iteration. At each iteration, left branch is reduced by a ratio $\cos(\alpha)$ and right branch is reduced by a ratio $\sin(\alpha)$. We therefore have two different homothety ratios, unless $\alpha = \frac{\pi}{2}$ but this correspond to the P_n Pythagoras tree we used before.

(For your information, $D_H(P_\alpha)$ is the solution of the equation $\cos(\alpha)^d + \sin(\alpha)^d = 1$.)

In this situation, the first way to handle fractal graphs explained in section 7.1 is no longer possible. Box covers order relationship used in the first step of the framing procedure can no longer be extracted from the fractal construction method. If we denote $k \times G$ the graph G such that all his weights edges are multiplied by k, we only have something like that:

$$\text{CBC}(\overline{P_{n+1,\alpha}},\varepsilon) \le \text{CBC}(\sin(\alpha)\times\overline{P_{n,\alpha}},\varepsilon) + \text{CBC}(\cos(\alpha)\times\overline{P_{n,\alpha}},\varepsilon) + 2$$

$$\text{DBC}(\overline{P_{n+1,\alpha}},\varepsilon) \ge \text{DBC}(\sin(\alpha)\times\overline{P_{n,\alpha}},\varepsilon) + \text{DBC}(\cos(\alpha)\times\overline{P_{n,\alpha}},\varepsilon)$$

Or in the general case for a fractal F with N homotheties with ratio r_i:

$$\text{CBC}(\overline{F_{n+1}},\varepsilon) \le \sum_{i=1}^{N} \text{CBC}(r_i\times\overline{F_n},\varepsilon) + a$$

$$\text{DBC}(\overline{F_{n+1}},\varepsilon) \ge \sum_{i=1}^{N} \text{DBC}(r_i\times\overline{F_n},\varepsilon) - a$$

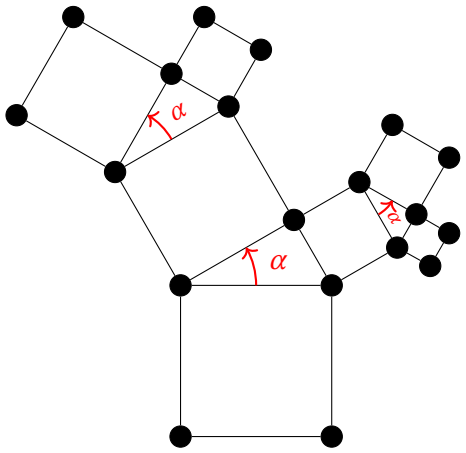

Fig. 41. $P_{3,\alpha}$ Asymmetrical Pythagoras Tree

with $a \ge 0$. It's particularly difficult to exploit this recurrence order relationship so our framing procedure doesn't seem to be the correct way to prove corollary Cor. 8.11.2 for all fractals with internal homotheties.

9 Testing

We created a graph framework in Java in order to implement CBC and DBC greedy algorithms and to verify fractal dimensions results. Feel free to visit the Gitlab project website page: https://gitlab.com/yzu-algorithm-lab/box-covering-algorithms/. In the following sections, we're gonna explain how the project is structured and what results it produced.

9.1 Overview

Inheritance is pretty straightforward. *Sierpinski* (for Sierpinski Triangle), *Pythagoras* (for Pythagoras Tree) and *Tetrahedron* (for Sierpinski Tetrahedron) classes all inherit from a *Fractal* class implementing all fractal graphs with internal homothety of the same ratio, itself inheriting from a *Graph* class.

DBC and *CBC* classes inherit from an abstract BC class for brute-force box-covering algorithms. *ApproxDBC* and *ApproxCBC* classes inherit from *DBC* and *CBC* and are used for box-covering approximation algorithms. Last but not least, *ApproxDBCRand* and *ApproxCBCRand* classes inherit from *ApproxDBC* and *ApproxCBC* and are used for randomizing box-covering approximation algorithms.

To these we add *Dijkstra* class, which performs distance calculations on an instance of the *Graph* class, and a *LinearRegression* class, which performs linear regressions. The final *ApproxDim* class takes as input a box-covering algorithm and a graph, executes the former on the latter for a given range and finally performs the linear regression to obtain the approximate value of the graph's fractal dimension. We'll talk about *ApproxDimAdjusted* and *ApproxDimRescaled* classes in section 9.4.

Other additional classes are designed to test the various classes by generating the fractals and returning the various *ApproxDim* results.

We show in Fig. 42 an overview of our framework (without test classes).

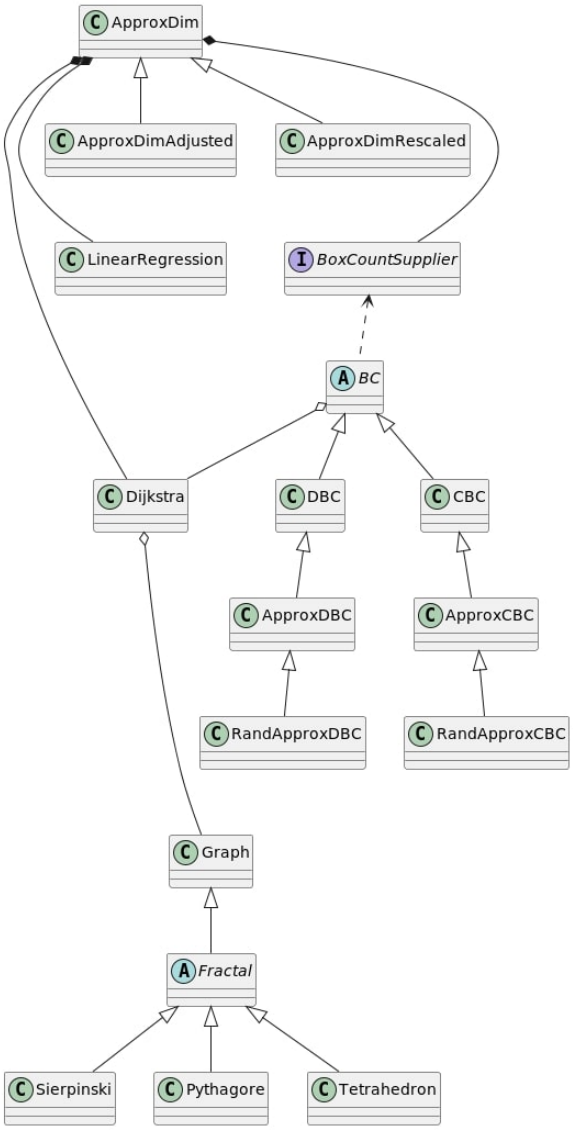

Fig. 42. Fractal graph framework's class diagram

9.2 Implementation choices

9.2.1 *Box-covering algorithms.*

Diamond problem. At the start of the implementation, we had the idea of making merged classes *ApproxBC* instead of *ApproxDBC* and *ApproxCBC* and *RandApproxBC* instead of *RandApproxDBC* and *RandApproxCBC*, exactly as in Fig. 43, but multiple inheritance is impossible is Java due to diamond problem.

As a reminder, the diamond problem in object-oriented programming arises in multiple inheritance scenarios, where a class inherits from two classes that both inherit from a common base class. This creates an ambiguity because the derived class has two copies of the base class, leading to potential conflicts in data and method resolution. Some programming languages, like C++, handle this by using virtual inheritance, while others, like Java, avoid the issue by disallowing multiple inheritance.

And we couldn't avoid the problems by using interfaces instead of classes because we had to use attributes from parent classes. As a direct consequence of this issue, code in *ApproxDBC* and *ApproxCBC* and in *RandApproxDBC* and *RandApproxCBC* are almost identical but impossible to factorize.

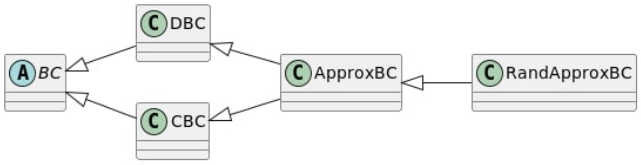

Fig. 43. Boxcovering package we first wanted to implement

Generally, what changes between different box-covering classes is the implementation of the program entry point method *boxCover()*.

Branch and bound algorithms. In *DBC* and *DBC*, *boxCover()* calls a branch and bound recursive function *boxCover(boxNum, bestBoxNum)*. We first go to any root of the choice tree (the choices being those of the box centers), which enables us to obtain a first cover for DBC or CBC, then we explore the neighboring branches, stopping if the current cover of DBC or CBC is worse than the registered cover, or updating the registered cover with the current cover if we arrive at another root with better cover. Although this approach prunes many branches of the choice tree, the algorithm remains exponential and tests can only be performed on very small graphs.

A simple execution of the branch and bound algorithm is shown in Fig. 44. On the left side, we've got a simple graph G with its optimal complete box cover with $\varepsilon = 1$ colored in blue ($CBC(G, \varepsilon) = 1$). On the right side, we've got its corresponding choice tree. Nodes correspond to box covers. Leaves colored in red correspond to complete box covers (in $\mathcal{C}(G, \varepsilon)$). We can see that according to CBC Heuristic, we first choose b vertex as complete box center. $\{b\}$ is a root so we stop there. We next explore $\{a\}$. $\{a\}$ is not a root but we don't go further because we already found a box-cover whose size is 1. Finally, we do the same for $\{c\}$ node. You can see that we visited 3 branches instead of 7 thanks to branch and bound algorithm.

Approximation algorithms. In *ApproxDBC* and *ApproxCBC*, *boxCover()* calls a recursive function *boxCover(boxNum)* which acts almost like algorithms detailed in Section 6. To be precise, each call to the recursive function corresponds to an iteration of the while loop in Algorithms 8 and 7. Instead of sorting every vertices at each recursive call like in *DBC* and *CBC*, we just need to get the best vertex thanks to the *selectBestVertex()* method. However, we don't use exactly the same Box Covering Heuristics explained in Section 6. Indeed, instead of $\deg_P(u)$, we use $w_P(u) = \sum_{v \in P | (u,v) \in E} w(u, v)$. On unweighted graphs, $\deg_P(u) = w_P(u)$.

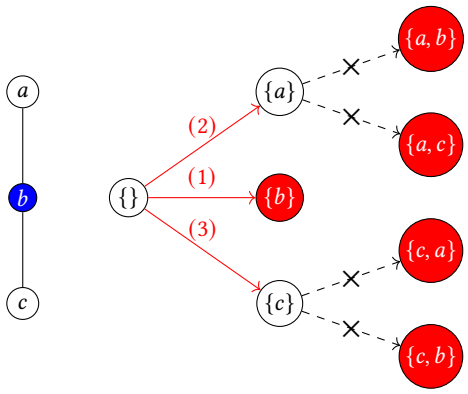

Fig. 44. On the left side: a simple graph. On the right side: the corresponding choice tree for branch and bound complete box-covering algorithm.

In *RandApproxDBC* and *RandApproxCBC*, *boxCover()* calls a recursive function *boxCover(boxNum)* whose goal is to execute the randomized approximation algorithm n times by modifying the function *selectBestVertex()*. These classes have a seed which updates at each approximation algorithm execution to make sure random choices are not the same. The method *getBoxCover()* returns the best cover found.

9.2.2 Fractal graph construction. On the other side, in order to construct fractal graph, the abstract *Fractal* class just needs a *minimalFractal()* method (which corresponds to first iteration of fractal graph), a *fractalNumVertices()* method (which provides the number of vertices of the fractal) and a *getRenamedList()* method (which, without going into too much technical detail, explains how to merge sub-fractal vertices at each iteration).

Sierpinski, *Pythagoras* and *Tetrahedron* classes contain nothing else that 3 implementations of these 3 methods, which a slight difference with *Pythagoras* which have to precise to *Fractal* class that it needs to add additional vertices during the merging process.

The *BoxCountSupplier* interface is just a way for us to provide any different box covering algorithm to *ApproxDim* class.

9.2.3 Dimension approximation. Finally, the linear regression we execute on our box cover results is based on the affine relationship between the epsilon and the box covers size natural logarithms. As shown in the introduction:

$$\ln(N(\varepsilon)) = \ln(c) - D_{\text{box}} \ln(\varepsilon)$$

So we just need to return the opposite of the slope of the linear regression as the fractal dimension approximation. Computing the intercept of a linear regression over the epsilon and the ratio of the epsilon and the box covers size natural logarithms (to simulate the limit when $\varepsilon \to 0^+$) isn't a great idea because the ratio doesn't converge fast enough to obtain realistic results.

We show in Fig. 45 a more detailed diagram class of our framework (without test classes).

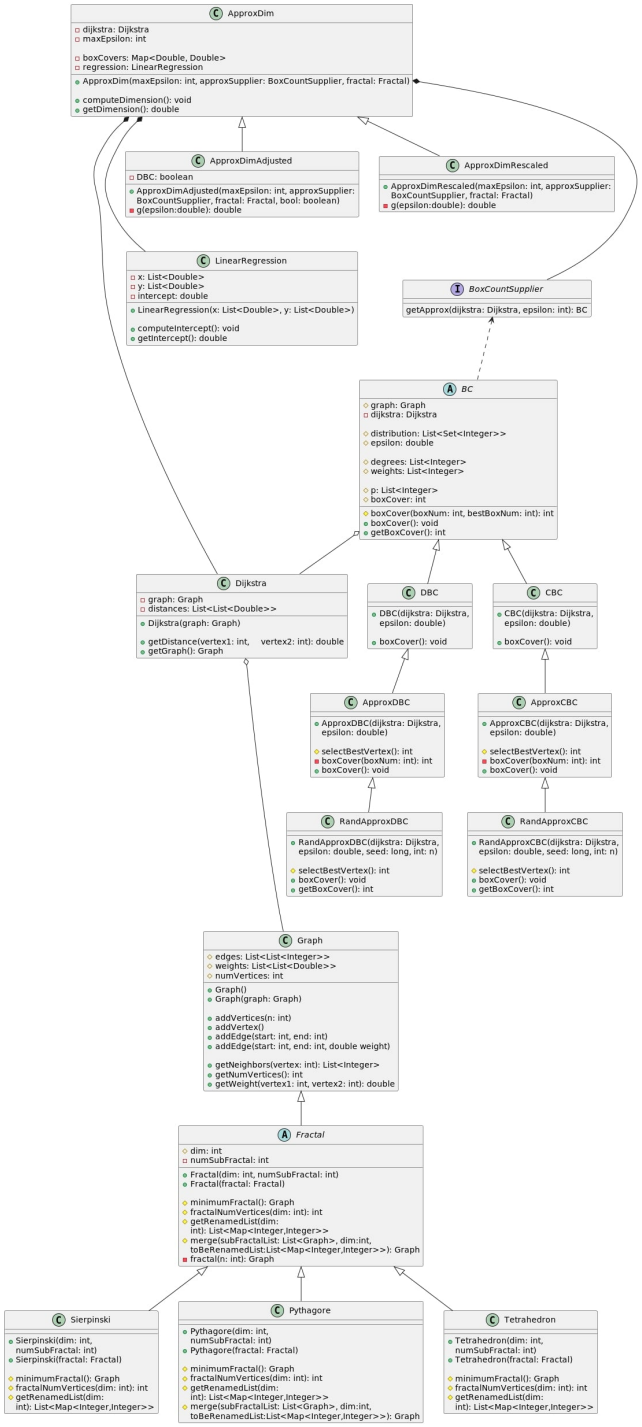

Fig. 45. Fractal graph framework's complete class diagram

9.3 Results

Tests were created to verify the proper functioning of *Graph*, *Dijkstra*, *LinearRegression*, *Fractal*, *Sierpinski*, *Pythagoras*, *Tetrahedron*

classes. Dimension results were a bit more complicated to generate, because of the complexity of the fractal generation. With higher fractal dimension, exponential time complexity of fractal generation goes way beyond time complexity of box-covering algorithms. That's why we created an other *Fractal* constructor taking as argument precedent fractal iterations rather that an integer dimension, in such a way as to resume work when previous iterations are merged, rather than from the very beginning. Unfortunately, even this measure can't stand against the exponential power of the number of vertices. For this reason, we usually are limited to execute box-covering algorithms on less than the ten first iterations of our fractal graphs (7 for the Sierpinski Triangle, 8 for the Pythagoras Tree and 5 for the Sierpinski Tetrahedron). An implementation in a more efficient programming language as C++ or Rust could be under consideration.

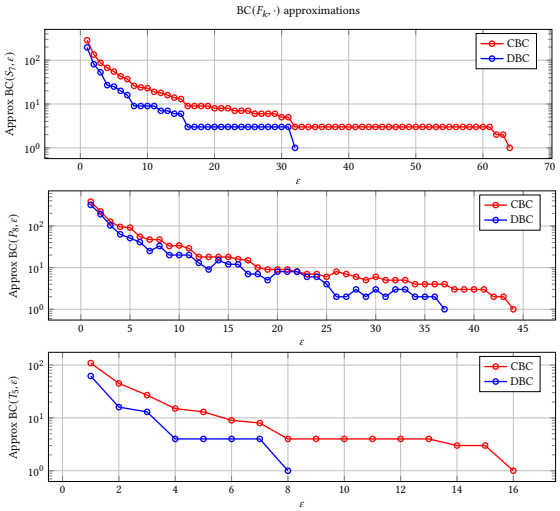

Fig. 46. Fractal graph box covers results

9.3.1 *Box-covering results.*

Basic algorithm. In Fig. 46, we can see the results for the execution of both complete and disjoint box-covering approximation algorithms (with a logarithmic scale to better see variations between the two box covers results). We stop computing box cover size when we get one whose size is 1 (it approximately corresponds to $\varepsilon_b^{n_b}(F_n)$) because all the following ε would also cover fractal vertices with only one box. First, we can notice that these results experimentally confirm Th. 2.1. CBC curve is always greater or equal to DBC. Second, we can observe that BC curves seem hyperbolic in $\frac{1}{\varepsilon^k}$ which experimentally confirms Th. 8.11.

Randomized algorithm. In Fig. 47, we can see the gain in percent for the execution of both complete and disjoint BC randomized approximation algorithms in comparison with normal approximation algorithms. Most of the time, relative gap between those two is negative for DBC algorithm and positive for CBC algorithm: these are consistent results, randomized algorithm always provide better

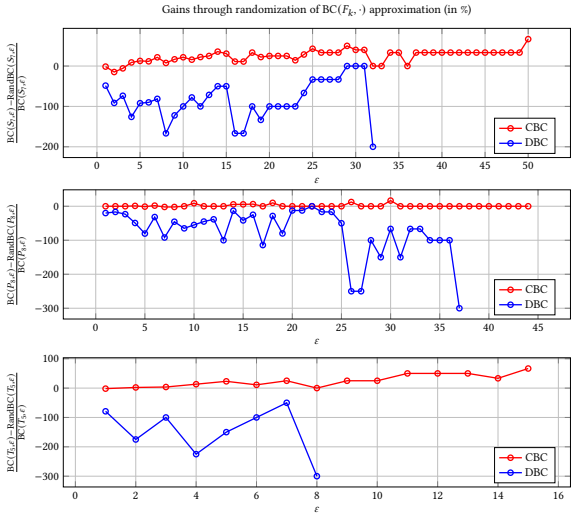

Fig. 47. Fractal graph randomized box covers gain

covers than the normal one. This means that randomize indeed added value to our BC experimentation.

Interesting fact: randomizing seems to be much more efficient with DBC than with CBC. This could be explained by the fact that choosing a wrong disjoint box center is more "punitive" than choosing a wrong complete box center in relation to their respective optimization (when we choose a disjoint box center, we have to remove more vertices from the potential center list than when we choose a complete box center).

We can also notice than relative gap seems to be reducing for DBC and increasing for CBC as ε is increasing. This could be directly linked to the complexity of the box center choice tree. With a small ε, it's complicated to quickly find a good box cover but with few vertices, randomized algorithm proportionally explores more branches of the choice tree. As a conclusion, it appears clear that randomizing algorithm works better.

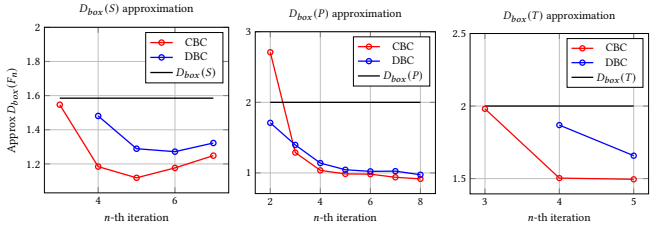

Fig. 48. Fractal graph dimension approximation results

9.3.2 *Fractal dimension results.*

Basic approximation. In Fig. 48, we can see fractal dimension approximation results with complete and disjoint BC approximation algorithms. Given the value sought, these approximations may seem disappointing. This can be explains by a lot of different factors. For example, we can see that the approximated dimension for the

Sierpinski Triangle is increasing towards the homothety dimension on the last iterations. We can suppose that it converges towards the homothety dimension but the convergence speed is too slow for our computers to be able to show the correspondence between the box-counting dimension and the homothety dimension.

On an other hand, our approximation algorithms tend to overestimate optimal CBC sizes and underestimate optimal DBC sizes. It's reasonable to assume that this implies that optimal CBC results slope is smaller than approximated CBC results slope and optimal DBC results slope is greater than approximated DBC results slope. If we apply a linear regression on these hypothetical results, it means that optimal fractal dimension values are between approximated fractal dimension values. Approximations results provide a "framing" for optimal results.

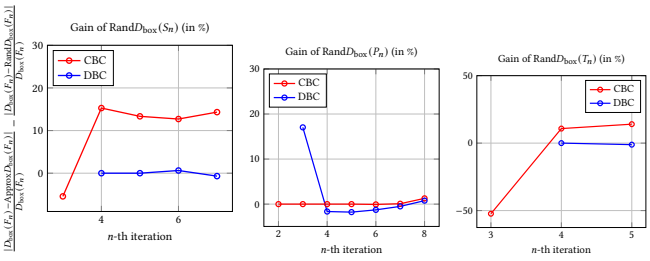

Fig. 49. Fractal dimension approximation with randomized box covers gain

Randomized approximation. In Fig. 49, we can see the gain in percent for the fractal dimension approximation using both complete and disjoint BC randomized approximation algorithms in comparison with normal approximation algorithms. We can notice that the difference between normal and randomized relative gaps is almost always positive. This means that randomized results are much closer to the real homothety dimension than normal results. In view of these results, we cannot deny that randomized algorithms are far more efficient than normal algorithms for fractal dimension computation.

9.4 Possible improvements

9.4.1 Adjusted dimension. We thought of using the knowledge of $\varepsilon_c^{n_c}(F_n)$ or $\varepsilon_d^{n_d}(F_n)$ to "readjust" the distribution of the ε. The idea was to multiply every ε by the ratio of $\varepsilon_c^{n_c}(F_n)$ or $\varepsilon_d^{n_d}(F_n)$ and the last ε used. Let's say ε^* is the first ε such that $\text{Approx}_{\text{CBC}}(F_n, \varepsilon)$=1 or $\text{Approx}_{\text{DBC}}(F_n, \varepsilon)$=1 (in our implementation, BC approximation algorithms execution stops when we found box covers with only 1 element). We want to compose ε by this function g:

$$\forall\ \varepsilon \leq \varepsilon^*,\ g(\varepsilon) = \varepsilon \times \frac{\varepsilon_c^{n_c}(F_n)}{\varepsilon^*} \text{ or } g(\varepsilon) = \varepsilon \times \frac{\varepsilon_d^{n_d}(F_n)}{\varepsilon^*}$$

and then execute the linear regression on this set of epsilons and box covers. Let's notice that $g(\varepsilon^*) = \varepsilon_c^{n_c}(F_n)$ or $g(\varepsilon^*) = \varepsilon_d^{n_d}(F_n)$. We have therefore readjusted ε^* such that it represent the correct ε in a perfect execution of box-covering algorithm. The postulate is that the distribution of approximate box covers keeps the same shape as the distribution of exact box covers, except for one scale.

This approach was implemented in *ApproxDimAdjusted* class which inherits from *ApproxDim* class. Before giving data to *LinearRegression* class, we compose each ε by g. Be aware that g definition varies according to the BC algorithm used so *ApproxDimAdjusted* class has an boolean field *DBC* indicating weather we're using a disjoint or complete BC algorithm.

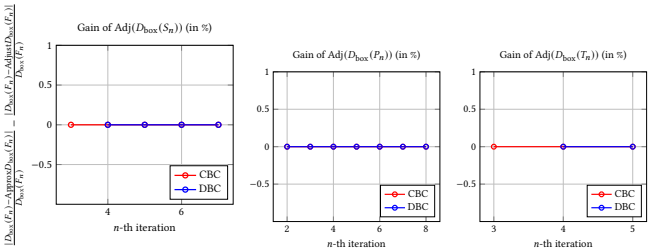

Fig. 50. Fractal dimension graph adjustment gain

In Fig. 50, we can see the gain in percent for the adjusted fractal dimension approximation using both complete and disjoint BC approximation algorithms in comparison with normal fractal dimension approximation. As you can see, the difference between normal and adjusted relative gaps is way to small to be observed with the naked eye (we're talking about a difference of the order of 10^{-16}). We can conclude dimension adjustment isn't efficient enough.

9.4.2 Rescaled dimension. On the other hand, we thought of rescaling the epsilon exactly like in demonstration of Cor. 8.11.2. Indeed, every ε-boxes in $\overline{F_n}$ is supposed to correspond to a $\frac{\varepsilon}{r^n}$-boxes in F_n.

This approach was implemented in *ApproxDimRescaled* class which inherits from *ApproxDim* class. Before giving data to *LinearRegression* class, we multiply each ε by r^n.

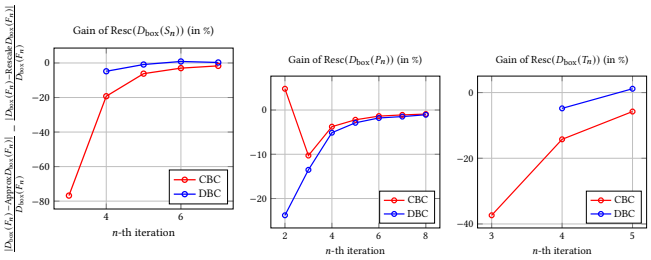

Fig. 51. Fractal dimension graph adjustment gain

In Fig. 51, we can see the gain in percent for the rescaled fractal dimension approximation using both complete and disjoint BC approximation algorithms in comparison with normal fractal dimension approximation. We can notice that the difference between normal and rescaled relative gaps is almost always negative. This means that rescaled results are further from to the real homothety dimension than normal results. Rescaling approach does not therefore seem very effective.

10 Conclusion

As a conclusion, we obtained a lot of information about box-covering algorithms:

1. We've established the order relationship between disjoint box covers and complete box covers and in which case we had equality.
2. We've reduced set covers problems to box covering problems in order to demonstrate their $\mathcal{NP}$-complexity.
3. We've proved disjoint box cover and complete box cover duality by their formulation into linear programs.
4. We've explained a few variants and what it changes in our considerations.
5. We've observed that none box-covering algorithms are approximations even with heuristics.
6. We've studied different ways to consider fractal graphs and how to rigorously define their dimension.
7. We've manipulated fractal graphs in order to find framing for box covers and then proved a specific case of Schroeder's conjecture.
8. We've implemented a framework in Java in order to compute fractal dimension approximations.

One exprimental observation that we haven't really touched in the testing section, but which confirms the relevance of our work, is that geometric fractals properties are easily found in fractal graphs. Most of the time (except for example for the Von Koch snow flakes), graphs can contain as much information as geometric figures. And to be honest, most of graph theory demonstrations in this paper could easily be transposed to geometric fractals.

It appears that dual box cover problems comparative study was far richer than it initially appeared. There's a good chance that with the right approach, results obtained here could be generalized to other more complicated fractal dimension definitions. It gives great hope for the future of fractal research.

11 Glossary of notations

Here are all the news notations from this paper used used in several sections / demonstrations.

- $\mathcal{D}(G,\varepsilon)$ — Disjoint Box Cover Set:
$$\{D \subseteq V \mid (\mathcal{B}(d,\varepsilon))_{d\in D} \text{ are pairwise disjoint. }\}$$
- $\mathcal{C}(G,\varepsilon)$ — Complete Box Cover Set:
$$\left\{C \subseteq V \mid V = \bigcup_{c\in C} \mathcal{B}(c,\varepsilon)\right\}$$
- $\overline{\mathcal{D}}(G,\varepsilon)$ — Optimal Disjoint Box Cover Set:
$$\{D \in \mathcal{D}(G,\varepsilon) \mid \mathrm{DBC}(G,\varepsilon) = |D|\}$$
- $\overline{\mathcal{C}}(G,\varepsilon)$ — Optimal Complete Box Cover Set:
$$\{C \in \mathcal{C}(G,\varepsilon) \mid \mathrm{CBC}(G,\varepsilon) = |C|\}$$
- $f_{D,C} : D \to \mathcal{P}(C)$ — Matching function between $D \in \mathcal{D}(G,\varepsilon)$ and $C \in \mathcal{C}(G,\varepsilon)$:
$$\forall\, d \in D,\ f_{D,C}(d) = \{c \in C \mid d \in \mathcal{B}(c,\varepsilon)\}$$
- $\overline{f_{D,C}} : D \to C$ — Bijective matching function (matching function which requires $|D| = |C|$):
$$\forall\, d \in D,\ \overline{f_{D,C}}(d) = !\, c \in C \mid d \in \mathcal{B}(c,\varepsilon)$$
- $\mathcal{U}(D,\varepsilon)$ — Uncovered vertices of $D \in \mathcal{D}(G,\varepsilon)$:
$$\left\{v \in V \mid \nexists\, d \in D \mid v \in \mathcal{B}(d,\varepsilon)\right\}$$
- $\mathcal{O}(C,\varepsilon)$ — Overcovered vertices of $C \in \mathcal{C}(G,\varepsilon)$:
$$\left\{v \in V \mid \exists\, c \neq c' \in C \mid v \in \mathcal{B}(c,\varepsilon) \cap \mathcal{B}(c',\varepsilon)\right\}$$
- $\mathcal{U}(\mathcal{D}(G,\varepsilon))$ — Uncovered vertices of (G,ε):
$$\bigcup_{D\in\overline{\mathcal{D}}(G,\varepsilon)} \mathcal{U}(D,\varepsilon)$$
- $\mathcal{O}(\mathcal{C}(G,\varepsilon))$ — Overcovered vertices of (G,ε):
$$\bigcup_{C\in\overline{\mathcal{C}}(G,\varepsilon)} \mathcal{O}(C,\varepsilon)$$
- $S_d(k)$ — A "step" of $\mathrm{DBC}(G,\cdot)$ function (non-empty if and only if $k \in \mathrm{DBC}(G,\mathbb{R}_+)$):
$$\{\varepsilon \in \mathbb{R}_+ \mid \mathrm{DBC}(G,\varepsilon) = k\}$$
- $S_c(k)$ — A "step" of $\mathrm{CBC}(G,\cdot)$ function (non-empty if and only if $k \in \mathrm{CBC}(G,\mathbb{R}_+)$):
$$\{\varepsilon \in \mathbb{R}_+ \mid \mathrm{CBC}(G,\varepsilon) = k\}$$
- n_d — Number of "steps" in $\mathrm{DBC}(G,\cdot)$ function:
$$\#\{k \in [\![1,|V|]\!] \mid \exists\, \varepsilon \in \mathbb{R}_+ \mid \mathrm{DBC}(G,\varepsilon) = k\}$$
- n_c — Number of "steps" in $\mathrm{CBC}(G,\cdot)$ function:
$$\#\{k \in [\![1,|V|]\!] \mid \exists\, \varepsilon \in \mathbb{R}_+ \mid \mathrm{CBC}(G,\varepsilon) = k\}$$
- $(\varepsilon_d^i)_{i\in[\![1,n_d]\!]}$ — sequence extracted from $\mathrm{DBC}(G,\cdot)$:
$$\varepsilon_d^i = \inf S_d(k_i)$$
- $(\varepsilon_c^i)_{i\in[\![1,n_c]\!]}$ — sequence extracted from $\mathrm{CBC}(G,\cdot)$:
$$\varepsilon_c^i = \inf S_c(k_i)$$
- $\mathcal{F}$ — Set of fractals with N internal homotheties of the same ratio r.
- F_n — Iteration of a fractal in $\mathcal{F}$ without r^n rescaling.
- $\overline{F_n}$ — Iteration of a fractal in $\mathcal{F}$ with r^n rescaling.
- $D_{\mathrm{box}}(F)$ — Box-counting dimension of $F \in \mathcal{F}$ (if and only if the limit exists):
$$\lim_{\varepsilon\to 0^+} \frac{\ln(\mathrm{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \lim_{\varepsilon\to 0^+} \frac{\ln(\mathrm{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$
- $D_{\overline{\mathrm{box}}}(F)$ — Upper box-counting dimension of $F \in \mathcal{F}$:
$$\overline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\mathrm{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \overline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\mathrm{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$
- $D_{\underline{\mathrm{box}}}(F)$ — Lower box-counting dimension of $F \in \mathcal{F}$:
$$\underline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\mathrm{CBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})} = \underline{\lim_{\varepsilon\to 0^+}} \frac{\ln(\mathrm{DBC}(F,\varepsilon))}{\ln(\frac{1}{\varepsilon})}$$
- $D_H(F)$ — Homothety dimension of $F \in \mathcal{F}$:
$$\frac{ln(N)}{ln(\frac{1}{r})}$$

References

[1] S. Akbari and S.J. Kirkland. On unimodular graphs. *Linear Algebra and its Applications*, 421, 2006.

[2] Barabási. Albert-László and Albert. Réka. Emergence of scaling in random networks. *Science*, 286(5439), 1999.

[3] Yanguang Chen. Equivalent relation between normalized spatial entropy and fractal dimension. *Physica A: Statistical Mechanics and its Applications*, 553, 2020.

[4] Gérard Cornuéjols, Michele Conforti, and Giacomo Zambelli. *Integer Programming*. Springer, 2014.

[5] George Datseris, Inga Kottlarz, Anton P. Braun, and Ulrich Parlitz. Estimating fractal dimensions: A comparative review and open source implementations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(10), 2023.

[6] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley & Sons, 2003.

[7] A. J. Hoffman and J. B. Kruskal. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems*, 38:223–246, 1956.

[8] Richard M Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

[9] J.S. Kim, K-I. Goh, B. Kahng, and D. Kim. Fractality and self-similarity in scale-free networks. *New Journal of Physics*, 9(6), 2007.

[10] Péter Tamás Kovács, Marcell Nagy, and Roland Molontay. Comparative analysis of box-covering algorithms for fractal networks. *Applied Network Science*, 6(1), 2021.

[11] Leling Kuang, Zengqiang Zhao, Fei Wang, Yong Li, Feifei Yu, and Zhiyuan Li. A differential evolution box-covering algorithm for fractal dimension on complex networks. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 693–699. IEEE, 2014.

[12] Mauro Locci, Gianluigi Concas, Roberto Tonelli, and Ilaria Turnu. Three algorithms for analyzing fractal software networks. *WSEAS Transactions on Information Science and Applications*, 7, 2010.

[13] B.B. Mandelbrot. *Fractals: Form, Chance, and Dimension*. W.H. Freeman and Co., San Francisco, 1977.

[14] B.B. Mandelbrot. *The Fractal Geometry of Nature*. W.H. Freeman and Co., 1982.

[15] Christian M. Schneider, Tobias A. Kesselring, José S. Andrade, and Hans J. Herrmann. Box-covering algorithm for fractal dimension of complex networks. *Physical Review E*, 86(1), 2012.

[16] M.R. Schroeder. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. W.H. Freeman and Co., 1991.

[17] Pavel Skums and Leonid Bunimovich. Graph fractal dimension and the structure of fractal networks. *Journal of Complex networks*, 8(4), 2020.

[18] Chaoming Song, Lazaros K Gallos, Shlomo Havlin, and Hernán A Makse. How to calculate the fractal dimension of a complex network: the box covering algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(03), 2007.

[19] Manuel Sorge. Tikz hypergraphs. https://gitlab.com/gruenwald/tikz-hypergraphs, 2019.

[20] Yuanyuan Sun and Yujie Zhao. Overlapping-box-covering method for the fractal dimension of complex networks. *Phys. Rev. E*, 89, 2014.

[21] Hongrun Wu, Li Kuang, Feng Wang, Qi Rao, Maoguo Gong, and Yuanxiang Li. A multiobjective box-covering algorithm for fractal modularity on complex networks. *Appl. Soft Comput.*, 61:294–313, 2017.

[22] Jingyou Wu. An information fractal dimensional relative entropy. *AIP Advances*, 14(02), 2024.

[23] Haicheng Zhang, Yufeng Hu, Xianliang Lan, Sankaran Mahadevan, and Yong Deng. Fuzzy fractal dimension of complex networks. *Applied Soft Computing*, 25:514–518, 2014.

[24] W Zheng, Q Pan, C Sun, YF Deng, XK Zhao, and Z Kang. Fractal analysis of mobile social networks. *Chinese Physics Letters*, 33(3), 2016.