1 VOUW: A Framework

The very first objective of this paper is to generalize our ideas into a theoretic framework. Such a framework provides us with the necessary tools to formally define the problem and the idealized space of solutions. The VOUW framework consists of two parts: (1) an concise definition that allows us to reason about what one can and cannot expect to find from a set of data and what this data could look like, and (2) a generalized encoding scheme that connects our framework to the MDL-principle.
De volgende twee alinea's moeten nog wat preciezer, ofwel negeer ze maar gewoon.

As previously introduced, the overall objective is to create a description D' of some dataset D that is both optimal in length and in the amount of semantically relevant information it provides on D. To this end we seek to derive a model H and a set of residuals $D|H$ (literally 'the data given the model') such that $D' = \{H, D|H\}$. Given that D' is optimal, there is exactly one D' for a given D and this D' is unique ($D \mapsto D'$ is bijective). Readers might already be familiar with this relation from the MDL-principle, and indeed, we will later use this principle to show that the *smallest* D' is the most optimal.

From now on, let us take D to be some matrix A. Its optimal model H_A should be able to capture all recurring structure, or patterns, in A that is semantically relevant, while containing nothing else. Informally we could say that H_A is the 'perfect summary' of A. In addition to H_A, there is the set of residuals $A|H_A$. These residuals can be thought of what is normally called the 'error' or 'noise' in A_H. Another, more meaningful, way to look at $A|H_A$ is to think of it as a mapping from H_A back to A. Indeed we will see that given $A|H_A$ we can unambiguously construct A from H_A. We will therefore call $A|H_A$ the set of (pattern) *instantiations* from this point on. In this light it is not hard to see that we are actually losslessly *compressing* A by finding a set of symbols (the model) H_A and encoding A using these symbols in $A|H_A$ (the instances).

1.1 Problem Domain

The VOUW framework is defined on grid-like data that is both bounded and discrete. This paper takes A to be a rectangular $M \times N$ matrix, but both the shape and dimensionality of A could in theory be different. A is defined as a typical matrix with elements $a_{i,j}$ where row i is on $[0; N)$ and column j is on $[0; M)$. Furthermore we say that $a_{i,j} \in \mathbb{N}$, e.g. is of the integral type, and that $0 \leq a_{i,j} < b$. b of A, written as $b(A)$, is said to be the *base* of A and this number basically tells us the number of different values A contains. $b = 2$ is a special case and makes A a *boolean matrix*. While it is strictly not imperative that the base is known beforehand, both the encoding and some data-specific operators can be greatly simplified if it is.

Now that we know what the domain of our data looks like, it is time to state what it is that we are looking for. Since we are mining for patterns in A it makes sense to informally say that a model H_A should be a set of 'patterns' over A. This set should only contain unique patterns and not more than absolutely necessary, otherwise it would not be optimal. Conversely the set of pattern instantiations must contain 'everything that is not a pattern'. But what exactly is a pattern? We will try to formalize these concepts in the next sections.

1.2 Patterns

We can encode a collection of elements $a_{i,j} \in A$ with a *spatial offset* (i, j) and a *magnitude offset.* A set of these offsets represents a *pattern* over A.

Definition 1. *A **pattern** is a set of pairs in the form* $(\boldsymbol{v}, w)$ *, where* $\boldsymbol{v}$ *is an n-dimensional spatial offset within* A *and scalar* w *is a magnitude offset.* $\boldsymbol{v}$ *is unique, i.e. it occurs at most once in a pattern.*

Definition 2. *The relation* $\leq$ *on pairs* $x, y \in X$ *is defined for a pattern* X *such that* $x \leq y$ *iff for* $\boldsymbol{v}_x = (i_x, j_x)$ *and* $\boldsymbol{v}_y = (i_y, j_y)$ *holds that* $2M + (N + i_x - 1)(2M - 1) + (M + j_x - 1) \leq (N + i_y - 1)(2M - 1) + (M + j_y - 1)$.

Theorem 1. *Any pattern* X *is a totally ordered set.*

Proof. We use the relation $\leq$ that we have defined on elements of X. Notice that X is a set of pairs but only the first element $\boldsymbol{v}$ is considered by the relation. By this definition we can already say that X is partially ordered: because $\boldsymbol{v} = (i, j)$ and j is bounded by M we can map (i, j) onto $\mathbb{N}$ by $n = (N+i-1)(2M-1)+(M+j-1)$. This gives us reflexivity, antisymmetry and transitivity as on $\mathbb{N}$ by the less-than-equal relation. We use the fact that $\boldsymbol{v}$ is unique in any given X to show that for $x_i, x_j \in X$ either $x_i \leq x_j$ or $x_j \leq x_i$ and thus satisfies comparability, making X totally ordered.

Definition 3. *Pattern* X *is a **singleton pattern** if* $|X| = 1$

The scalar w in each pair $(\boldsymbol{v}, w)$ is the magnitude offset. Let us define what we exactly mean by that. Imagine we create a singleton pattern $X = ([0, 0], 2)$ of elements in A. X contains one element that has magnitude 2. So simply said, it matches all elements $a_{i,j} = 2$. In this case the magnitude offset is taken from zero, the lowest possible value in A. Now let us define a pattern Y:

$$Y = \big\{X_0 + ([0, 0], 0), X_1 + ([1, 0], 1), X_2 \ldots\big\}$$

In this notation we define the $+$ operator as $X + \omega = \{x + \omega | x \in X\}$. Let us take $X_{0,1}$ to be X. Clearly, the pattern Y contains all elements of X at their origin and offset by $([1, 0], 1])$. The magnitude offset is in this case relative to X and yields $Y = \{([0, 0], 2), ([1, 0], 3)\}$, matching all horizontal adjacent elements $2, 3$ in A. Now assume that such a set of elements actually occur in A starting from $\boldsymbol{p} = (x, y)$. These elements are then said to be the *image* of Y at $\boldsymbol{p}$. We call such a vector $\boldsymbol{p}$ a *pivot.* A pivot is just another offset that shifts the center $(0, 0)$ of the pattern by some (x, y).

Definition 4. *The* ***image*** *in A of a pattern X at pivot $\boldsymbol{p}$ is the set of elements $a_{i,j} \in A$ such that $(i,j) = \boldsymbol{v} + \boldsymbol{p}$ for each $(\boldsymbol{v}, w) \in X$.*

Recall that H_A is a set of patterns over A, but which patterns are there exactly? For this we define $\tilde{\mathcal{P}}_A$ to be the set of all patterns over A. It contains the singleton patterns of A all the way up to the most complex pattern (i.e., $A \in \tilde{\mathcal{P}}_A$). While this is certainly useful, consider the following definition.

Definition 5. *Pattern X is* ***isomorphic*** *to pattern Y if there exists some pivots $\boldsymbol{p}, \boldsymbol{q}$ for which the image of X at $\boldsymbol{p}$ equals the image of Y at $\boldsymbol{q}$. We write $X \cong Y$.*

Another important observation is that for a pattern X to occur in A, its values not need be strictly equal to the values of the corresponding elements $a_{i,j}$, as long as it has the same meaning in the context of the data. We call this *semantic equivalence.*

Definition 6. *Patterns X and Y are* ***semantically equivalent*** *in A if there exists some t for which $X \circ t \cong Y$. We write $X \equiv Y$.*

We call $X \circ t$ a *variant* and $\circ$ some relation on a set of elements of A that transforms X to Y using t. We restrict $\circ$ in such way that it cannot add or remove elements from X and that any variant $X \circ t$ is unique.

Let us take for example the strings $(1,0,0,1)$ and $(0,1,1,0)$ from some dataset. While having a completely different absolute value, one could say that there are equivalent if we define $X \circ t$ as $\forall_{x \in X} \cdot x + t \mod B$. We can easily see that $(1,0,0,1) \circ 1 = (0,1,1,0) \circ 0$. The variants here are 1 and 0 respectively. Another example is the equivalence of the vectors $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $(1,1)$. In this case the relation $\circ$ would be equal to t times their transpose. Many different relations and combinations thereof are possible, as long a they are semantically meaningful for the data in A. The elegance in this is that we have abstracted this domain-specific knowledge away into one single operator.

It would seem that we have now two different equivalence relations defined on patterns: isomorphism and (semantic) equivalence. The set $\tilde{\mathcal{P}}_A$ contains both isomorphic patterns as well as semantically equivalent patterns. It is not hard to see that this would lead to unnecessary complexity. For all isomorphic and equivalent patterns we would like to pick one unambiguous representative.

Definition 7. *Pattern X is in* ***canonical form*** *if for all elements $x_i = (\boldsymbol{v}_i, w_i)$ hold that $v_0 = (0,0)$ and $v_i \leq v_{i+1}$.*

Theorem 2. *If X and Y are in canonical form, $X \cong Y$ if and only if $X = Y$.*

Theorem 3. *For any pattern X not in canonical form, there exists a pattern X' in canonical form.*

For equivalence it is not possible to pick a single representative because the ordering of elements in a pattern is only defined using spatial offset and not by magnitude. Furthermore, the relation $\circ$ is a black box to us. We know, however, that any given pattern has a finite number of variants because $\circ$ cannot add or remove elements and the set of values of $a_{i,j}$ is finite. Therefore we can pick as representative a finite set of equivalent patterns.

Definition 8. *The **equivalence set** X^* of canonical pattern X_0 is the set of all canonical patterns $X_0, X_i, \ldots$ such that holds $X_i \equiv X_{i+1}$.*

Theorem 4. *If some pattern $X_i \in X^*$ then $X_i* = X^*$, i.e. the set X^* is complete and unique.*

This is still not good enough as we want to be able to enumerate the elements of X^*. We therefore define another total ordering on X^* as follows: We know that every pattern in $X_i \in X^*$ is in canonical form and thus the order of the elements in these patterns is fixed. $\circ$ tells us that the number of elements is also finite, say it is n. We now can represent each $X_i \in X^*$ as a number $k = k_0 k_1 \ldots k_n$ of n digits in base b, one for each element $x_{ij} \in (X_i)$ such that $k_j = \omega_j$. However if elements are just repositioned by $\circ$ this number k may not be unique. We therefore order all patterns with equal k by mapping them onto another n-digit number $l_0, \ldots, l_n$ but this time using $l_j = \boldsymbol{v}_j$. To order the vectors $\boldsymbol{v}$ we define $\leq$ as in Definition 2. Lastly we define $(X^*)_a \leq (X^*)_b$ as $k_a \leq k_b \vee l_a \leq l_b$. This gives us a partial ordering while the requirement on variants to be unique fulfils comparability and gives us a total ordering on X^*.

The ordering of X^* conveniently lets us enumerate all patterns equivalent to X.

Definition 9. *X_0^* is the pattern in X^* such that no pattern $(X^*)_i \leq X_0^*$ exists. We define X_t^* as the t-th pattern in X^* such that $X_t^* = (X_0^*) \circ t$.*

Using X_0^* we define another, more useful set of patterns over A. We say that the set $\mathcal{P}_A$ is the set of all patterns X_0^* over A. A special set $\mathcal{P}_A^0$ is the set of all X_0^* with length 1 (thus sets of singleton patterns).

1.3 Regions

With the definitions given above we can almost formalize the definition of model H_A and its instantiations. Because we want a model to only contain unique patterns, we say that any $H_A \subseteq \mathcal{P}_A$. As mentioned before, patterns can occur any number of times in A at different pivots. We call a tuple of a pivot, a pattern and variant a *region*.

Definition 10. *A **region** is a tuple in the form $(X, t, \boldsymbol{p})$ such that $X \in \mathcal{P}_A$ and the image X_t^* occurs at $\boldsymbol{p}$ in A. The image of a region $(X, t, \boldsymbol{p})$ is defined as equal to the image of pattern X_t^* at pivot $\boldsymbol{p}$.*

Indeed a region is just the smallest amount of information required to map a pattern onto its image in A. Unsurprisingly the set of instantiations to H_A can be defined as a set of regions:

Definition 11. *The instantiations $\{A|H_A\}$ of H_A is a set of regions $\{R_1, R_i, \ldots, R_N\}$ such that the union of the images $\bigcup_{i=1}^{N} \texttt{image}(R_i)$ equals A.*

1.4 Constructing Patterns and Regions

As patterns are the building blocks of our model we would like to be able to have an intuitive way to construct increasingly large patterns. One way to do this is to define some join operation on two patterns such that the result is a new pattern containing elements from both. A simple union would not work in this case, as this would disregard any spatial relation those two patterns might have. We therefore have to define a special 'pattern union' that takes a third argument, namely the distance between the two operands.

Definition 12. *The union of patterns X and Y with distance $\boldsymbol{\delta}$ is defined as $X \cup \{y_i + (\boldsymbol{\delta}, 0) \mid y_i \in Y\}$ for any X, Y such that $X \cap Y = \emptyset$. We write $X \bigcup^{\delta} Y$.*

We can immediately see that $\bigcup^{\delta}$ is a more restricted version of $\bigcup$. Because we take $\boldsymbol{\delta}$ to be the distance from X to Y where X is the origin, properties such associativity and commutativity do not hold for $\bigcup^{\delta}$. Another concern is that if X and Y are in canonical form, is $X \bigcup^{\delta} Y$ too? The answer is, fortunately, yes (though we will not show it here).

The union of patterns has its counterpart for the set of instantiations. We can define a special sum on two regions (recall a region is a tuple not a set) by internally using the pattern union we just defined.

Definition 13. *The sum of regions $R = (X, t, \boldsymbol{p})$ and $S = (Y, u, \boldsymbol{q})$ is defined as $R + S = (Z * 0, t', \boldsymbol{p})$ where $Z = (X \circ t) \bigcup^{\delta} (Y \circ u)$, $\boldsymbol{\delta} = \boldsymbol{q} - \boldsymbol{p}$ and $(Z * 0) \circ t' = Z$.*

This is in fact a quite complex definition and it is also the fundamental operation of VOUW. Again we take the first operand as the origin such that $R + S \neq S + R$. Before the pattern union is computed between the patterns of R and S, notice how the variants are calculated first. The resulting pattern will have a different structure/number of elements and will thus belongs to a different equivalent set - which is unknown. This is circumvented by taking $Z * 0$ rather than Z to be the pattern of $R + S$. We use both this and the $*$ operator to find that Z is the t-th element in $Z*$. Finally we take this t' to be the variant of $R + S$.

Notice that the region sum has the same limitation as the pattern union: two regions can only be summed if they do not overlap, or in other words, their patterns need to be disjunct (after applying variants and $\boldsymbol{\delta}$). While this is a serious limitation, we will show in the next section that it is not of any practical relevance.

1.5 The Set $\mathcal{H}_A$ and $\bar{\mathcal{H}}_A$

Now all the pieces are ready to define the set of all possible models of A and the set of all possible instantiations to models of A. We will write these sets as $\mathcal{H}_A$ and $\bar{\mathcal{H}}_A$ respectively. To this end will first take H_A^0 to be the model with only singleton patterns (patterns of length 1). Note that as all unique singleton patterns are required to create a singleton model of A and therefore $H_A^0 = \mathcal{P}_A^0$.

The instantiations to H_A^0 are the set $A|H_A^0$. This is the set of regions that map one singleton pattern on every a_{ij}. Because every a_{ij} is only the image of exactly one variant of a patterns in H_A^0, there is only one way to do this.

Theorem 5. *Given a matrix A and the set of unique patterns $\mathcal{P}_A$, H_A^0 and $A|H_A^0$ are also unique.*

Now we inductively define the set $\bar{\mathcal{H}}_A$ of all instantiations of all models over A:
Base case: $A|H_A^0 \in \bar{\mathcal{H}}_A$
By induction: If $\bar{h}$ is in $\bar{\mathcal{H}}_A$ then for any $R, S \in \bar{h}$, the set $h \setminus \{R, S\} \cup \{R + S\}$ is also in $\bar{\mathcal{H}}_A$.

Notice how the addition of two regions is the fundamental operation here. Indeed we can add any two regions together in any order and eventually this results in just one big region: A. The elegance in this is that by this inductive definition, the regions R and S never overlap and thus their sum is always valid. By the same principle this sum is always a pattern with an image in A.

The construction of instance sets also implicitly defines the corresponding models. While this may seem odd - defining models for instantiations instead of the other way around - note that there is no unambiguous way to find a instance set for a given model. Instead we find the following trivial definition by applying the inductive construction rule above.

Definition 14. *The set $\mathcal{H}_A$ of all models over A is the set such that $\mathcal{H}_A = \left\{ \{X \mid (X, t, \boldsymbol{p}) \in h\} \;\middle|\; h \in \bar{\mathcal{H}}_A \right\}$.*

So given any instance set $\bar{h} \in \bar{\mathcal{H}}_A$ there is a corresponding set in $\mathcal{H}_A$ of all patterns that occur in $\bar{h}$. This finalizes the definition of models and instantiations.

Theorem 6. *Given any instance set $\bar{h} \in \bar{\mathcal{H}}_A$ and its corresponding model in $\mathcal{H}_A$, the matrix A can be reconstructed unambiguously.*

1.6 The Optimal Model

We have now found *all* possible models for A and have defined a nice method to construct both instance sets and models inductively. While this is certainly useful, we still need to find *the optimal* model. To this end we make the connection with the MDL-principle that says, informally, by minimizing the following equation the given model is optimal.

$$L_1(H_A) + L_2(A|H_A)$$

Here the functions L_1, L_2 are two independent length functions that make up the coding scheme for two-part MDL. Recall that we essentially want to compress the data in A. We therefore need to find an encoding scheme that encodes both model and instantiations lossless and without redundancy.

Say we have some set of symbols $S = \{S_0, S_1, \ldots, S_n\}$ and each symbol has a length $l_0, l_1, \ldots, l_n$. These symbols could be for example a code word for each pattern in a model. We now want to find lengths l_i using the Kraft-inequality such that holds that $\sum_{i=0}^{N} r^{-l_i} \leq 1$. In this case we say that $r = 2$, because there are two symbols (0 and 1) in our code alphabet so we can measure code lengths in bits. The Kraft-inequality gives us a bijection between code lengths and probability distributions. This is one of the main ideas of the MDL-principle. So in our example we can write a probability distribution $P(X), X \in H_A$ as the chance of X occurring in our instance set. We can then use Shannon's Entropy $l_i = -\log(P(X_i))$ to compute the exact number of bits a pattern should optimally be encoded with.

Notice how common code words are shorter then rarely used code words. According to the Kraft-inequality this gives us an optimal code. The number of bits we compute are however, real numbers and not integers. While this does certainly not result in a practical encoding, we do not actually need to encode the data as we are only interested in its hypothetical length.

1.7 Encoding Models and Instantiations (niet lezen)

Even though we will not actually be encoding A, we want the calculation of the encoded length to be as precise as possible. The simple reason for this is that the only information that we provide to the MDL equation and thus the optimization process, is this encoded length. A coarse granularity may lead to unwanted results and overfitting. While this may sound inexact we will make it more precise by using Shannon's Entropy to reason about the optimal length for any element we encode.

From the definitions of $\mathcal{H}_A$ and $\bar{\mathcal{H}}_A$ we know that patterns both occur as elements in H_A as well as in the regions of $A|H_A$. It makes sense that we encode the complete patterns once for the models and then refer to them from each region by a code word. This makes the practical materialisation of the model a *code table* that maps each pattern to some code word. So how do we generate these code words? Recall that we are only interested in the theoretical length of a code and not so much in the actual word itself. To compute the optimal length we first and foremost need to find out what the chance is that a certain pattern occurs and this in turn depends on the prevalence in the dataset.

Definition 15. *Given a set of instantiations $A|H_A$, we take* $\texttt{usage}(X)$ *of a pattern X to be the prevalence of X^* in $A|H_A$. More precisely*

$$\texttt{usage}(X) = |\{R \mid R = (Y, t, \boldsymbol{p}) \in \{A|H_A\}, Y^* = X^*\}|$$

From this definition we see that the *usage* of a pattern is sum a of how often any of its variants occur as an instance. Using this function we find the probability that a certain pattern occurs simply by $P(X) = \frac{|\{A|H_A\}|}{\texttt{usage}(X)}$. The optimal length of a code word L^C can then be found by Shannon's Entropy. Now that we know the length of a code word the pattern itself must also be encoded. Recall that a pattern consists of pairs of spatial and magnitude offsets. In a $M \times N$ matrix there can be at most $4(M-1)(N-1)$ distinctive spatial offsets and this number can be used to obtain the minimum number of bits each offset can be encoded with. The magnitude offset depends on the number of possible values in A, which is $b(A)$. Assuming the distribution of values in A is uniform, we can also use a fixed value here. This brings the length of an encoded pattern to:

$$L(X) = |X| \cdot \Big(-\log\Big(\big(4(M-1)(N-1)\big)^{-1}\Big) - \log\Big(b(A)^{-1}\Big)\Big)$$

This leaves only the instantiations still to be encoded, which form a list rather than a table. Each region simply refers to its pattern X by using the code word we computed earlier and we already know its length. The pivot is again a fixed number that depends on the total number of possible pivots - MN in this case. Encoding the variant is harder because we have never clearly defined $|X^*|$. In principle the upper bound for the number of variants for a given pattern X is $b(A)^{|X|}$, since we established X^* is at least finite. This is not a practical figure however, given that there probably are far less elements in A. A solution is to limit the total number of variants for X to the number that *we know about* at a given moment. We can find this number in a similar way to the `usage` function.

Definition 16. *Given a set of instantiations $A|H_A$, we define*

$$\texttt{variants}(X) = |\{Y \circ t \mid R = (Y, t, \boldsymbol{p}) \in \{A|H_A\}, Y^* = X^*\}|$$

Which basically defines $\texttt{variants}(X)$ as the number of distinct variants of X that occur within $A|H_A$. In a similar way as the encoded length of a pattern, we can now define the encoded length of region.

$$L(R) = -log\big(MN^{-1}\big) - \log\big(\texttt{variants}(X)^{-1}\big) + L^C(X)$$

Where X is the pattern in R. We can now compute the lengths of both region and patterns as well as the length of a code word for each pattern. By summing of all patterns and regions we can finally give the total length of the model and the set of instantiations.

$$L(H_A) = \sum_{X \in H_A} L^C(X) + L(X)$$
$$L(A|H_A) = \sum_{R \in A|H_A} L(R)$$

2 VOUW: An Algorithm

Now that we have finalized the theoretical foundation we can begin to formulate the problem and work towards an algorithmic solution. Before we do so, however, we take a moment to sketch the complexity of the current solution space. In order to do this we use the same steps as the inductive definition of $\bar{\mathcal{H}}_A$ as given in section 1.5. We start with $\bar{h} = \bar{H}_A^0$, which by definition contains exactly MN regions. We then pick two arbitrary regions and replace these with their sum to obtain $\bar{h}'$. Simple combinatorics say we can do this in $\binom{MN}{2}$ ways. Since now $|\bar{h}'| = MN - 1$, we have $\binom{MN-1}{2}$ possibilities for the next step, and so forth until we have only one big region left that contains exactly A (the completely overfit model). We can simplify this sequence to

$$\prod_{n=0}^{MN-2} \binom{MN-n}{2} = \prod_{n=0}^{MN-2} \frac{MN-n}{2(MN-n-2)!} = \frac{(MN)!(MN-1)!}{2^{MN-1}}. \quad (1)$$

While we must stress that this is an upper bound in the worst case that no pattern ever occurs more than once, the complexity class this naive approach yields is still quite impossible. In the remainder of the section we will describe a baseline algorithm that utilizes heuristics and additional constraints to reduce this complexity to close to linear.

2.1 Heuristics

It appears that it is not feasible to search the entire solution space due to the combinatorial explosion we have just seen. We will now describe a practical baseline algorithm that can search the solution space in close to linear time, if we are willing to find 'a good solution' rather than 'the optimal solution'. The baseline algorithm can be used to show complexity and give guarantees on convergence, while we will later modify it for practical applications by adding decomposition (a form of pruning) and support for domain-specific equivalence.

The first step is to add two additional constraints to the search:

1. The search must be **greedy**: when we unite two regions, we will never go back to a state where they are separate.
2. Two regions can only be united if they are **adjacent**.

The first constraint is very straightforward. The greedy approach reduces the complexity considerably with the danger of converging on local minima. While this sounds problematic, notice that the discussed models are already an approximation. Therefore it is hard to justify the computational expenses looking for 'the optimal approximation'. Furthermore, we will introduce an additional method later on that decreases the probability of convergence on local minima.

For the second constraint we must first define what we mean by *adjacency*.

Definition 17. *Two regions $R = (X, t, \boldsymbol{p})$ and $S = (Y, u, \boldsymbol{q})$ with $(i, j) = \boldsymbol{q} - \boldsymbol{p}$ are* ***adjacent*** *if*

$$\boldsymbol{p} < \boldsymbol{q} \wedge j \leq \texttt{colMax}(X) + 1 \wedge j \geq \texttt{colMin}(X) - 1 \wedge i \leq \texttt{rowMax}(X) + 1.$$

For $\boldsymbol{p} < \boldsymbol{q}$ we use the same ordering for two-dimensional vectors as in Definition 2. This ensures that the position of R is always before S. This solves problem the that we could consider both $X \cup^{\delta} Y$ and $Y \cup^{-\delta} X$ as candidates while these are isomorphic. We define the function `adjacent`(R) to be set of all regions adjacent to R. Figure **??** depicts more clearly what such a set looks like.

In order to compute complexity later, we would like to know the size of this set of adjacent regions. Fortunately it is trivial to give an upper bound for some region $R = (X, t, \boldsymbol{p})$.

$$|\texttt{adjacent}(R)| = \big(\texttt{width}(X) + 1\big) \times \big(\texttt{height}(X) + 1\big) - |X| + 1 \qquad (2)$$

According to this, the smallest possible region whose image is a single element in A has at most 4 adjacent regions. For the rest where complexity is concerned, we only need to know that the number of adjacent regions is on average roughly $O(|X|^{1.3})$.

2.2 The baseline algorithm

Now that we have defined and motivated the heuristics we use, it is time to discuss the algorithm itself. We will with an outline given by Listing 1.

Algorithm 1 vouw

```
$h \leftarrow H_A^0$
$\bar{h} \leftarrow \bar{H}_A^0$
repeat
    $C \leftarrow \emptyset$                                          ▷ Set of candidates
    $\forall_{c \in C}\ usage(c) = 0$          ▷ Relation $C \rightarrow \mathbb{Z}$ stores occurrence per candidate
    for all $R = (X, p) \in \bar{h}$ do          ▷ Pattern $X$ with pivot $p$ in the instance set
        for all $S = (Y, q) \in \texttt{adjacent}(R)$ do
            $\delta \leftarrow q - p$
            $c \leftarrow (X, Y, \delta)$                              ▷ Candidate tuple
            $C = C \cup c$
            $usage(c) = usage(c) + 1$
        end for
    end for
    $c_{best} = (X, Y, \delta) \in C : \forall_{c \in C}\ \texttt{gain}(c) \leq \texttt{gain}(c_{best})$     ▷ Select best candidate
    $g = \texttt{gain}(c_{best})$
    if $g > 0$ then
        $h = h \cup (X \cup^{\delta} Y)$                  ▷ Add the union pattern to the model
        for all $R \in \bar{h} \mid R = R_{c_{best}}$ do
            for all $S \in \texttt{adjacent}(R) \mid S = S_{c_{best}}$ do
                $\bar{h} = \bar{h} \cap R, S \cup R + S$            ▷ Replace R and S with their sum
            end for
        end for
    end if
until $g \leq 0$
```

In the algorithm described here we have omitted the variants and equivalence sets completely for the sake of simplicity. While in further sections we will gradually make it more precise, we will now briefly focus on the overall structure. Similar to the inductive definition of the instance set (see Section 1.5), the algorithm iteratively tries to find the most promising two patterns to combine. To this end we first look at all possible candidates: tuples (X, Y, δ) of two patterns X, Y and a distance δ. We then greedily take 'the best' candidate. This mechanism can be divided into five steps: (1) find candidates, (2) estimate candidates' value for `usage`(), (3) compute candidates' gain, (4) merge patterns $X \cup^{\delta} Y$ with the highest gain and (5) replace all matching regions with their sums.

The key to all these steps is knowing which candidate is the most promising. By estimating the `usage`() function (Definition 15) for the union pattern $X \cup^{\delta} Y$ we already gain insight into how useful a certain combination of patterns is: patterns that occur more often tend to reveal more structure and therefore compress the underlying data better. This is only part of the truth, however. When we for example combine two zero entries in a sparse matrix, this yields a pattern with high usage but we learn nothing about the data that we did not know already. To this end we define *gain*, the number of bits the two-part MDL equation $L(H) + L(D|H)$ decreases when we would pick a certain candidate.

Definition 18. *We define* `gain` *of candidate* $c = (X, Y, \delta)$ *such that:*

$$\begin{aligned}\texttt{gain}(X, Y, \delta) = & L(h) + L(\bar{h}) \\ & - L(h \cup \{X \cup^{\delta} Y\}) \\ & - L(\bar{h} \cup \{R + S | R, S \in \bar{h}, X = X_R, Y = X_S, \delta = p_S - p_R\} \\ & \quad \setminus \{R, S \in \bar{h} | X = X_R, Y = X_S, \delta = p_S - p_R\})\end{aligned}$$

In the definition above we slightly abused notation to let X_R and p_R denote pattern in region R and pivot of region R, respectively.