Chapter 6

Boolean functions and evasiveness

This chapter is devoted to a fascinating application to a computer science search problem involving "evasiveness," originally due to Forman [**67**]. Along the way, we will introduce Boolean functions and see how they relate to simplicial complexes and the evasiveness problem. For another interesting use of discrete Morse theory in the type of application discussed in this chapter, see [**59**].

6.1. A Boolean function game

Let's play a game[1]. For any integer $n \geq 0$, we'll pick a function f whose input is an $(n+1)$-tuple of 0s and 1s, usually written in vector notation as $\vec{x}$. The output of f is either 0 or 1. In other words, we are looking at a function of the form $f : \{0,1\}^{n+1} \to \{0,1\}$ with $f(\vec{x}) = 0$ or 1. Such a function is called a **Boolean function**. The **hider** creates a strategy for creating an input $\vec{x} := (x_0, x_1, \ldots, x_n)$. The **seeker** does not know the hider's input, and has to guess the output by asking the hider to reveal any particular input value (i.e., entry of the input vector). The seeker may only ask yes or no questions of the form "Is $x_i = 0$?" or "Is $x_i = 1$?" The chosen Boolean function f is known to both players. The object

[1] Not the kind that Jigsaw plays, thank goodness.

of the game is for the seeker to guess the output in as few questions as possible. The seeker wins if she can correctly predict the output without having to reveal every single input value. The seeker is said to have a **winning strategy** if she can always predict the correct output without revealing every input value, regardless of the strategy of the hider. The hider wins if the seeker needs to ask to reveal all the entries of the input vector.

In the following examples and exercises, it may help to actually "play" this game with a partner.

Example 6.1. Let $n = 99$ and let $f : \{0,1\}^{100} \to \{0,1\}$ be defined by $f(\vec{x}) = 0$ for all $\vec{x} \in \{0,1\}^{100}$. Then the seeker can always win this game in zero questions; that is, it does not matter what strategy the hider picks for creating an input since the output is always 0.

Example 6.2. Let $P_3^4 : \{0,1\}^{4+1} \to \{0,1\}$ by $P_3^4(x_0, x_1, x_2, x_3, x_4) = x_3$. The seeker can always win in one question by asking "Is $x_3 = 1$?" In general, we define the **projection** function $P_i^n : \{0,1\}^{n+1} \to \{0,1\}$ by $P_i^n(\vec{x}) = x_i$ for $0 \leq i \leq n$.

Exercise 6.3. Prove that the seeker can always win in one question for the Boolean function P_i^n.

Note that in Example 6.2, it is true that the seeker could ask an unhelpful question like "Is $x_4 = 1$?" but we are trying to come up with the best possible strategy.

Exercise 6.4. Let $P_{j,\ell}^n : \{0,1\}^{n+1} \to \{0,1\}$ be defined by $P_{j,\ell}^n(x_0, \ldots, x_n) = x_j + x_\ell \bmod 2$ for $0 \leq j \leq \ell \leq n$. Prove that the seeker has a winning strategy and determine the minimum number of questions the seeker needs to asks in order to win.

Although we can think of the above as simply a game, it is much more important than that. Boolean functions are used extensively in computer science, logic gates, cryptography, social choice theory, and many other applications. See for example [**131**], [**92**], and [**154**]. You can imagine that if a computer has to compute millions of outputs of Boolean functions, tremendous savings in time and memory are possible if the computer can compute the output by processing only minimal

input. As a silly but illustrative example, consider the Boolean function in Example 6.1. This Boolean function takes in a string of one hundred 0s and 1s and will always output a 0. If a computer program had to compute this function for 200 different inputs, it would be horribly inefficient for the computer to grab each of the 200 inputs from memory and then do the computation, because who cares what the inputs are? We know the output is always 0. In a similar manner, if we had to compute P_3^{999} on 1000 inputs, all we would need to do is call the third entry of each input, not all 1000 entries of each input. Hence, we can possibly save time and many resources by determining the minimum amount of information needed in order to determine the output of a Boolean function.

On the other hand, a Boolean function that requires knowledge of every single input value is going to give us some trouble. A Boolean function $f : \{0,1\}^{n+1} \to \{0,1\}$ for which there exists a strategy for the hider to always win is called **evasive**. In other words, such a function requires knowledge of the value of every single entry of the input in order to determine the output.

Define $T_2^4 : \{0,1\}^{4+1} \to \{0,1\}$ by $T_2^4(x_0, x_1, x_2, x_3, x_4) = 1$ if there are two or more 1s in the input and 0 otherwise. Let's simulate one possible playing of this game.

SEEKER: Is $x_1 = 1$?

HIDER: Yes.

SEEKER: Is $x_3 = 1$?

HIDER: Yes.

SEEKER: Now I know the input has at least two 1s, so the output is 1. I win!

While the seeker may have won this game, the hider did not use a very good strategy. Letting $x_3 = 1$ on the second question guaranteed two 1s in the input, so the output was determined. Rather, the hider should have had a strategy that did not reveal whether or not the input had two 1s until the very end. Let's try again.

SEEKER: Is $x_1 = 1$?

HIDER: Yes.

SEEKER: Is $x_3 = 1$?

HIDER: No.

SEEKER: Is $x_2 = 1$?

HIDER: No.

SEEKER: Is $x_0 = 1$?

HIDER: No.

SEEKER: Is $x_4 = 1$?

HIDER: Yes.

SEEKER: Now I know that the output is 1, but I needed the whole input. I lose.

Hence, the hider can always win if the Boolean function is T_2^4. In other words, T_2^4 is evasive.

Problem 6.5. The **threshold function** $T_i^n = T_i : \{0,1\}^{n+1} \to \{0,1\}$ is defined to be 1 if there are i or more 1s in the input and 0 otherwise. Prove that the threshold function is evasive.

6.2. Simplicial complexes are Boolean functions

Although our game may seem interesting, what does it have to do with simplicial complexes? There is a natural way to associate a simplicial complex to a Boolean function that satisfies an additional property called monotonicity.

Definition 6.6. Let $f : \{0,1\}^{n+1} \to \{0,1\}$ be a Boolean function. Then f is called **monotone** if whenever $\vec{x} \leq \vec{y}$, we have $f(\vec{x}) \leq f(\vec{y})$.

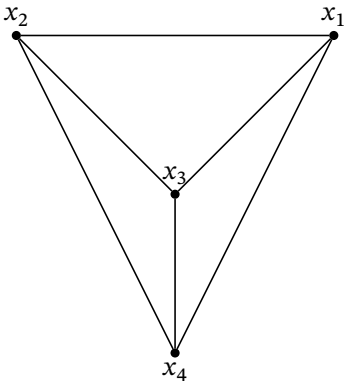
Exercise 6.7. Prove that the two constant Boolean functions $\mathbf{0}(\vec{x}) = 0$ and $\mathbf{1}(\vec{x}) = 1$ are monotone.

Exercise 6.8. Prove that the threshold function $T_i^n : \{0,1\}^{n+1} \to \{0,1\}$ defined in Problem 6.5 is monotone.

As we will see, the condition of being monotone will ensure that the following construction is closed under taking subsets. The construction is attributed to Saks et al. [**95**].

Definition 6.9. Let f be a monotone Boolean function. The **simplicial complex** Γ_f **induced by** f is the set of all sets of coordinates (excluding $(1,1,\dots,1)$) such that if $\vec{x} \in \{0,1\}^{n+1}$ is 0 exactly on these coordinates, then $f(\vec{x}) = 1$. We adopt the convention that if $x_{i_0} = \cdots = x_{i_k} = 0$ are the 0 coordinates of an input, then the corresponding simplex in Γ_f is denoted by $\sigma_{\vec{x}} := \{x_{i_0}, \dots, x_{i_k}\}$.

Example 6.10. You proved in Exercise 6.8 that the threshold function is monotone. Hence, it corresponds to some simplicial complex. Let us investigate $T_2^3 = T_2$. A simplex in Γ_{T_2} corresponds to the 0 coordinates of a vector $\vec{x}$ such that $T_2(\vec{x}) = 1$. We thus need to find all $\vec{x}$ such that $T_2(\vec{x}) = 1$. For example, $T_2((1,0,0,1)) = 1$. This has a 0 in x_1 and x_2, so this vector corresponds to the simplex x_1x_2. Also, $T_2((0,1,1,1)) = 1$, which corresponds to the simplex x_0. Finding all such simplices, T_2 produces the simplicial complex



Exercise 6.11. Why do we exclude vector $(1,1,\dots,1)$ in Definition 6.9?

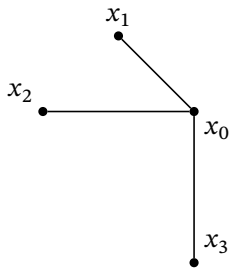
As mentioned above, we need monotonicity to ensure that Γ_f is a simplicial complex.

Problem 6.12. Show that the Boolean function $f : \{0,1\}^{n+1} \to \{0,1\}$ defined by $f(\vec{x}) = x_0 + x_1 + \cdots + x_n \bmod 2$ is not monotone. Then attempt to construct the corresponding Γ_f and show why it is not a simplicial complex.

Exercise 6.13. Compute the simplicial complexes $\Gamma_{\mathbf{0}}$ and $\Gamma_{\mathbf{1}}$ induced by the constant functions defined in Exercise 6.7.

Problem 6.14. Show that the projection $P_i^n : \{0,1\}^{n+1} \to \{0,1\}$ defined by $P_i^n(\vec{x}) = x_i$ is monotone, and then compute $\Gamma_{P_i^n}$.

Problem 6.15. Find a Boolean function f such that Γ_f is the simplicial complex below:



We now show that monotone Boolean functions and simplicial complexes are in bijective correspondence. In other words, a monotone Boolean function and a simplicial complex are two sides of the same coin.

Proposition 6.16. Let $n \geq 0$ be a fixed integer. Then there is a bijection between monotone Boolean functions $f : \{0,1\}^{n+1} \to \{0,1\}$ and simplicial complexes on $[n]$.

Proof. Let $f : \{0,1\}^{n+1} \to \{0,1\}$ be a monotone Boolean function. If $f(\vec{x}) = 1$ with 0 coordinates $x_{i_0}, \ldots, x_{i_k}$, write $\sigma_{\vec{x}} := \{x_{i_0}, \ldots, x_{i_k}\}$ and let Γ_f be the collection of all such $\sigma_{\vec{x}}$. To see that Γ_f is a simplicial complex, let $\sigma_{\vec{x}} \in \Gamma_f$ and suppose $\sigma_{\vec{y}} := \{y_{j_0}, y_{j_1}, \ldots, y_{j_\ell}\}$ is a subset of $\sigma_{\vec{x}}$. Define $\vec{y}$ to be 0 exactly on the coordinates of $\sigma_{\vec{y}}$. We need to show that $f(\vec{y}) = 1$. Suppose by contradiction that $f(\vec{y}) = 0$. Since $\sigma_{\vec{x}} \in \Gamma_f$, $f(\vec{x}) = 1$. Observe that since $\sigma_{\vec{y}} \subseteq \sigma_{\vec{x}}$, $\vec{y}$ can be obtained from $\vec{x}$ by switching all the coordinates in $\sigma_{\vec{x}} - \sigma_{\vec{y}}$ from 0 to 1. But then if $f(\vec{y}) = 0$, we have switched our inputs from 0 to 1 while our output has switched from 1 to 0, contradicting the assumption that f is monotone. Thus $f(\vec{y}) = 1$ and Γ_f is a simplicial complex.

Now let K be a simplicial complex on $[n]$. Let $\sigma \in K$ with $\sigma = \{x_{i_0}, \dots, x_{i_k}\}$. Define $\vec{x}_\sigma \in \{0,1\}^{n+1}$ to be 0 on coordinates $x_{i_0}, \dots, x_{i_k}$ and 1 on all other coordinates. Define $f : \{0,1\}^{n+1} \to \{0,1\}$ by $f(\vec{x}_\sigma) = 1$ for all $\sigma \in K$ and 0 otherwise. To see that f is monotone, let $f(\vec{x}_\sigma) = 1$ with 0 coordinates $x_0, \dots, x_k$ and suppose $\vec{x'}$ has 0 coordinates on a subset S of $\{x_0, \dots, x_k\}$. Since K is a simplicial complex, $S \in K$. Hence the vector with 0 coordinates S has output 1. But this vector is precisely $\vec{x'}$.

It is clear that these constructions are inverses of each other, hence the result.

□

Problem 6.17. Let $T_i^n : \{0,1\}^{n+1} \to \{0,1\}$ be the threshold function. Compute $\Gamma_{T_i^n}$.

6.3. Quantifying evasiveness

By Proposition 6.16, there is a bijective correspondence between monotone Boolean functions and simplicial complexes. Hence any concept we can define for a monotone Boolean function can be "imported" to a simplicial complex. In particular, we can define what it means for a simplicial complex to be evasive. Furthermore, we will generalize the notion of evasiveness by associating a number to the complex, quantifying "how close to being evasive is it?"

Let Δ^n be the n-simplex on vertices $v_0, v_1, \dots, v_n$. Suppose $M \subseteq \Delta^n$ is a subcomplex known to both players. Let $\sigma \in \Delta^n$ be a simplex known only to the hider. The goal of the seeker is to determine whether or not $\sigma \in M$ using as few questions as possible. The seeker is permitted questions of the form "Is vertex v_i in σ?" The hider is allowed to use previous questions to help determine whether to answer "yes" or "no" to the next question. The seeker uses an algorithm A based on all previous answers to determine which vertex to ask about. Any such algorithm A is called a **decision tree algorithm**.

Example 6.18. Let us see how this game played with simplicial complexes is the same game played with Boolean functions by starting with the Boolean function from Example 6.1, defined by $f : \{0,1\}^{99+1} \to \{0,1\}$ where $f(\vec{x}) = 0$ for all $\vec{x} \in \{0,1\}^{99+1}$. Then $n = 99$ so that we have Δ^{99}.

The subcomplex M known to both players is then the simplicial complex induced by the Boolean function, which in this case is again Δ^{99}. Now think about this. It makes no difference at all what hidden face $\sigma \in \Delta^{99}$ is. Since $\sigma \in \Delta^{99}$ and $M = \Delta^{99}$, σ is most certainly in M. So the seeker will immediately win without asking any questions, just as in Example 6.1.

Example 6.19. Now we will illustrate with the threshold function T_2^3, a more interesting example. In this case, $\Delta^n = \Delta^3$ and $M \subseteq \Delta^3$ is the simplicial complex from Example 6.10, the complete graph on four vertices, denoted by K_4, or the 1-skeleton of Δ^3. We can consider the same mock versions of the game as in the dialogue preceding Problem 6.5.

SEEKER: Is $x_1 \in \sigma$?

HIDER: No.

SEEKER: Is $x_3 \in \sigma$?

HIDER: No.

SEEKER: Now I know that $\sigma \in M$ because everything that is left over is contained in the 1-skeleton, which is precisely M.

Again, as before, this illustrates a poor choice of strategy by the hider (following the algorithm "always say no"[2]). Let's show a better strategy for the hider.

SEEKER: Is $x_1 \in \sigma$?

HIDER: No.

SEEKER: Is $x_3 \in \sigma$?

HIDER: Yes.

SEEKER: Is $x_2 \in \sigma$?

HIDER: Yes.

SEEKER: Is $x_0 \in \sigma$?

HIDER: No.

SEEKER: Now I know that $\sigma = x_2x_3 \in M$, but it is too late. I needed the whole input.

[2]Much to the displeasure of Joe Gallian.

Notice that the hider keeps the seeker guessing until the very end. If you follow along, you realize that everything depends on the answer to the question "Is $x_0 \in \sigma$?"

Denote by $Q(\sigma, A, M)$ the number of questions the seeker asks to determine if σ is in M using algorithm A. The **complexity** of M, denoted by $c(M)$, is defined by

$$c(M) := \min_A \max_\sigma Q(\sigma, A, M).$$

If $c(M) = n+1$ for a particular M, then M is called **evasive**, and it is called **nonevasive** otherwise. For the examples above, we saw that Δ^{99} is nonevasive while K_4 is evasive. Of course, this language is intentionally consistent with the corresponding Boolean function language. Furthermore, notice how evasiveness is just the special case where $c(M) = n+1$, while for nonevasive complexes $c(M)$ can have any value from 0 (extremely nonevasive) to n (as close to being evasive as possible without actually being evasive).

Any σ such that $Q(\sigma, A, M) = n + 1$ is called an **evader** of A. In Example 6.19, we saw that the hider had chosen $\sigma = x_2x_3$, so x_2x_3 is an evader. However, the hider could have just as easily answered "yes" to the seeker's last question, and in that case $\sigma = x_2x_3x_0$ would have been an evader. Hence, evaders come in pairs in the sense that by the time the seeker gets down to question $n + 1$, she is trying to distinguish between two simplices σ_1 and σ_2. If $\sigma_1 < \sigma_2$ with $\dim(\sigma_1) = p$, we call p the **index** of the evader pair $\{\sigma_1, \sigma_2\}$.

Exercise 6.20. Show that for a pair of evaders $\{\sigma_1, \sigma_2\}$, we have $\sigma_1 < \sigma_2$ (without loss of generality), $\dim(\sigma_2) = \dim(\sigma_1) + 1$, and that $\sigma_1 \in M$ while $\sigma_2 \notin M$. What does this remind you of?[3]

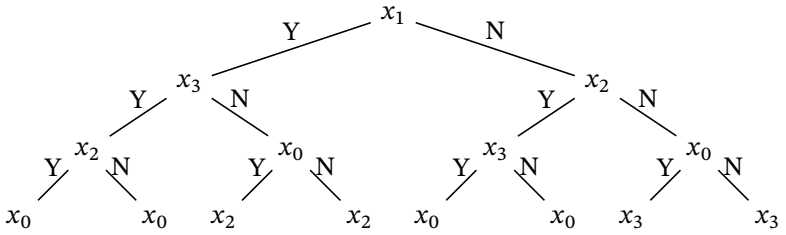
[3] I suppose in theory this could remind you of anything, from Sabellianism to pop tarts, but try to give a coherent answer.

6.4. Discrete Morse theory and evasiveness

You may have observed in Exercise 6.20 that a pair of evaders is a lot like an elementary collapse. Yet while a collapse represents something simple, a pair of evaders represents something complex, almost like a critical simplex. We will use discrete Morse theory to prove the following:

Theorem 6.21. Let A be a decision tree algorithm and let $e_p(A)$ denote the number of pairs of evaders of A having index p. For each $p = 0, 1, 2, \ldots, n-1$, we have $e_p(A) \geq b_p(M)$ where $b_p(M)$ denotes the pth Betti number of M.

In order to prove Theorem 6.21, we need to carefully investigate the relationship between the decision tree algorithm A chosen by the seeker and discrete Morse theory. To do this, let us explicitly write down a possible decision tree algorithm A that the seeker could execute in the case of our running Example 6.19. A node x_i in the tree below is shorthand for the question "Is vertex $x_i \in \sigma$?"



Once the seeker is on the very last row, she knows whether or not the previous vertices are in σ but cannot be sure if the current vertex is in σ. For example, suppose the seeker finds herself on the third value from the left, x_2. Working our way back up, the seeker knows that $x_0x_1 \in \sigma$ but isn't sure if $x_2 \in \sigma$. In other words, she cannot distinguish between x_0x_1 and $x_0x_2x_1$. The key to connecting this algorithm to discrete Morse theory is to use the observation of Exercise 6.20 and view $(x_0x_1, x_0x_2x_1)$ as a vector of a gradient vector field. In fact, we can do this for all nodes in the last row to obtain a gradient vector field on Δ^3. Actually, we have to be a little careful. Notice that the rightmost node on the bottom row

corresponds to "vector" $(\emptyset, x_3)$, which we do not consider part of a gradient vector field. But if we throw it out, we obtain a gradient vector field that collapses Δ^3 to the vertex x_3.

Problem 6.22. Write down explicitly a possible decision tree algorithm for the Boolean projection function $P_3^4 : \{0,1\}^{4+1} \to \{0,1\}$. (Note that this requires you to translate this into a question of simplicial complexes.) Compute the corresponding induced gradient vector field on Δ^4.

In general, any decision tree algorithm A induces a gradient vector field $V_A = V$ on Δ^n as follows: For each path in a decision tree, suppose the seeker has asked n questions and has determined that $\alpha \subseteq \sigma$. The $(n+1)$st (and final) question is "Is $v_i \in \sigma$?" Let $\beta = \alpha \cup \{v_i\}$, and include $\{\alpha, \beta\} \in V$ if and only if $\alpha \neq \emptyset$. We saw in Section 5.2.2 both a partial and a total order on the simplices of a simplicial complex. In the proof of the following lemma, we will construct another total order on the simplices of Δ^n.

Lemma 6.23. Let A be a decision tree algorithm. Then $V = V_A$ is a gradient vector field on Δ^n.

Proof. By Theorem 2.51, it suffices to show that V is a discrete vector field with no closed V-paths. That V is a discrete vector field is clear, since each path in the decision tree algorithm corresponds to a unique simplex.

To show that there are no closed V-paths, we put a total order $\prec$ on the simplices of Δ^n and show that if $\alpha_0, \beta_0, \alpha_1, \beta_1, \ldots$ is a V-path, then $\alpha_0 \succ \beta_0 \succ \alpha_1 \succ \beta_1 \succ \cdots$. That this is sufficient is Exercise 6.24. Assign to each edge labeled Y the depth of its sink node (lower node). For each path from the root vertex to a **leaf** (vertex with a single edge), construct a tuple whose entries are the values assigned to Y each time an edge labeled Y is traversed. In this way, we associate to each simplex $\alpha^{(p)}$ a tuple $n(\alpha) := (n_0(\alpha), n_1(\alpha), \ldots, n_p(\alpha))$ where $n_i(\alpha)$ is the value of the ith Y answer and $n_0(\alpha) < \cdots < n_p(\alpha)$. For any two simplices $\alpha^{(p)}$ and $\beta^{(q)}$ we define $\alpha \succ \beta$ if there is a k such that $n_k(\alpha) < n_k(\beta)$ and $n_i(\alpha) = n_i(\beta)$ for all $i < k$. If there is no such k and $q > p$, then $n_i(\alpha) = n_i(\beta)$ for all $0 \leq i \leq p$, so set $\alpha \succ \beta$. This order is shown to be transitive in Problem 6.25.

Now we show that the total order we defined above is preserved by a V-path. To that end, suppose $\alpha_0^{(p)}, \beta_0^{(p+1)}, \alpha_1^{(p)}$ is a segment in a V-path. We show that $\alpha_0^{(p)} \succ \beta_0^{(p+1)} \succ \alpha_1^{(p)}$. Since $(\alpha_0, \beta_0) \in V$, we have that $\alpha_0 \neq \alpha_1$ and $\alpha_1 \subseteq \beta_0$. Now α_0 and β_0 differ by only a single vertex, a difference which was not determined until the very bottom row of the decision tree. Hence, by definition of our total order, $n_i(\alpha) = n_i(\beta)$ for all $1 \leq i \leq p$ while $n_{p+1}(\beta) = n + 1$ and $n_{p+1}(\alpha_0)$ is undefined. Thus $\alpha \succ \beta$.

It remains to show that $\beta_0^{(p+1)} \succ \alpha_1^{(p)}$. Write $\beta_0 = u_0 u_1 \cdots u_p u_{p+1}$. For $\sigma = \alpha_0$ or β_0, we then have that the question $n_i(\beta_0)$ is "Is $u_i \in \sigma$?" Since $\alpha_1 \subseteq \beta_0$, there is a k such that $\alpha_1 = u_0 u_1 \cdots u_{k-1} u_{k+1} \cdots u_{p+1}$. Observe that the first $n_k(\beta_0) - 1$ questions involve $u_0, \ldots, u_{k-1}$ as well as vertices not in β_0, which also means that they are not in α_0 or α_1. Hence, the first $n_k(\beta_0) - 1$ answers will all be the same whether $\sigma = \alpha_0, \alpha_1$, or β_0; i.e., the corresponding strings all agree until n_k. For n_k, the question is "Is $u_k \in \sigma$?" If $\sigma = \beta_0$, the answer is "yes." If $\sigma = \alpha_1$, the answer is "no." Thus, the next value in the string for α_1 will be larger than the value just placed in $n_k(\beta_0)$. We conclude that $n_i(\alpha_1) = n_i(\beta_0)$ for $i < k$ and $n_k(\alpha_1) > n_k(\beta_0)$, hence $\beta_0 \succ \alpha_1$. □

It may aid understanding of the construction in Lemma 6.23 to draw an example and explicitly compute some of these strings.

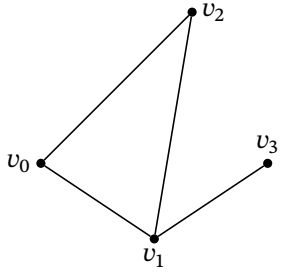
Problem 6.24. Suppose that there is a total order $\prec$ on Δ^n such that if $\alpha_0, \beta_0, \alpha_1, \beta_1, \ldots$ is a V-path, then $\alpha_0 \succ \beta_0 \succ \alpha_1 \succ \beta_1 \succ \cdots$. Prove that V does not contain any closed V-paths.

Problem 6.25. Prove that the order defined in Lemma 6.23 is transitive.

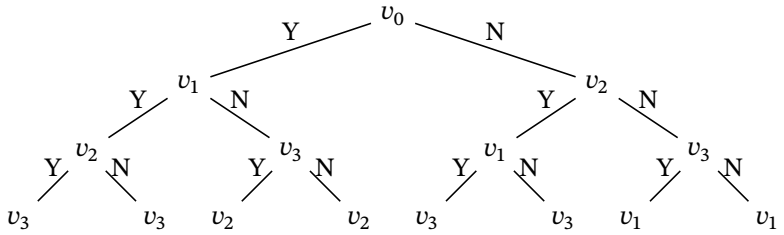
Exercise 6.26. Let A be a decision tree algorithm and V the corresponding gradient vector field. How can you tell which vertex V collapses onto immediately from the decision tree A?

There is one more fairly involved result that we need to combine with Lemma 6.23 in order to prove Theorem 6.21. The upshot is that after this result, not only will we be able to prove the desired theorem, but we will also get several other corollaries for free. We motivate this result with an example.

Example 6.27. Let $M \subseteq \Delta^3$ be given by



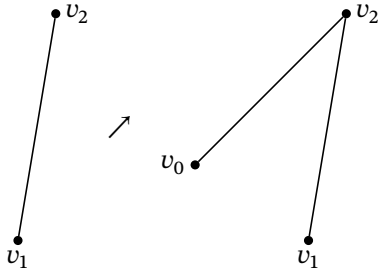
and consider the decision tree



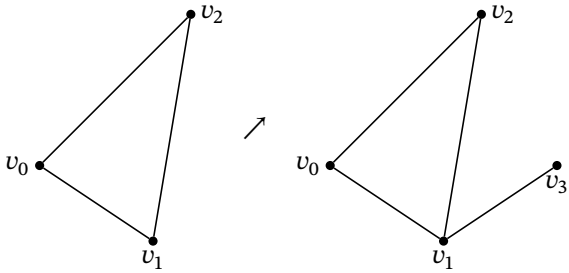
We know from Lemma 6.23 that A induces a gradient vector field on Δ^3, which in this case is a collapse onto v_1. Now the pairs of evaders are easily seen to be $\{v_0v_1, v_0v_1v_3\}, \{v_1v_2, v_1v_2v_3\}$, and $\{v_2, v_2v_3\}$, and by definition one element in each pair lies in M while the other lies outside of M. Hence, if we think of the evaders as critical elements, note that we can start with v_1 and perform elementary expansions and additions of the three evaders in M to obtain M. Furthermore, starting from M, we can perform elementary expansions and attachments of the three evaders not in M to obtain Δ^3. What are these elementary expansions? They are precisely the elements of the induced gradient vector field which are not the evaders. In the decision tree above, these are $\{v_0v_1v_2, v_0v_1v_2v_3\}, \{v_0v_3, v_0v_3v_2\}, \{v_0, v_0v_2\}$, and $\{v_3, v_3v_1\}$. Note that the excluded $\{\emptyset, v_1\}$ tells us where to begin. Explicitly, the series of expansions and attachments of evaders is given by starting with the single vertex

Next we attach the evader v_2:

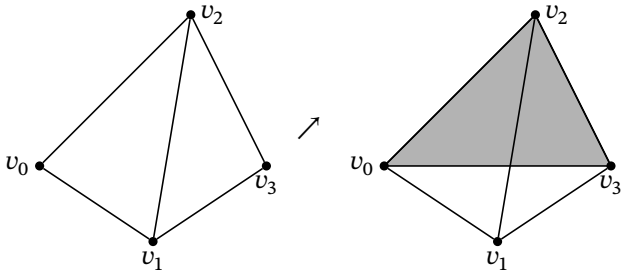
Now we attach the evader v_1v_2 and perform the elementary expansion $\{v_0, v_0v_2\}$:



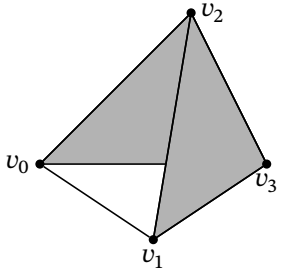
To obtain M, we attach the evader v_0v_1 and make the expansion $\{v_3, v_3v_1\}$:



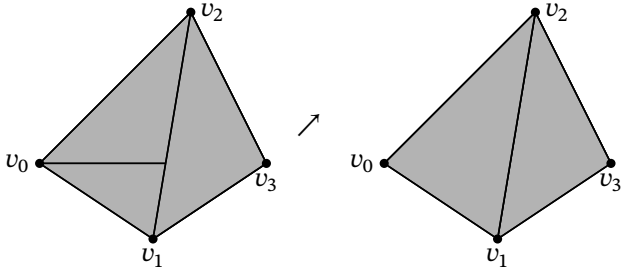
We continue to build from M all the way up to Δ^3. This begins with the attachment of the evader v_2v_3 followed by the expansion $\{v_0v_3, v_0v_3v_2\}$:



Next we have the attachment of evader $v_1v_3v_2$:



And finally we have the attachment of $v_0v_1v_3$ followed by the expansion $\{v_0v_1v_2, v_0v_1v_2v_3\}$:



This completes the construction of Δ^3 from A.

We now take the ideas from Example 6.27 and generalize to construct Δ^n out of expansions and attachments of evaders using the information in a decision tree A.

Theorem 6.28. Let A be a decision tree algorithm and k the number of pairs of evaders of A. If $\emptyset$ is not an evader of A, then there are evaders $\sigma_1^1, \sigma_1^2, \dots, \sigma_1^k$ of A in M, and evaders $\sigma_2^1, \sigma_2^2, \dots, \sigma_2^k$ of A not in M, along with a nested sequence of subcomplexes of Δ^n,

$$v = M_0 \subseteq M_1 \subseteq \cdots \subseteq M_{k-1} \subseteq M_k \subseteq M = S_0 \subseteq S_1 \subseteq \cdots \subseteq S_k \subseteq \Delta^n$$

where v is a vertex of M which is not an evader of A, such that

$$\begin{aligned}
v &\nearrow M_1 - \sigma_1^1 \\
M_1 &\nearrow M_2 - \sigma_1^2 \\
M_2 &\nearrow M_3 - \sigma_1^3 \\
&\vdots \\
M_{k-1} &\nearrow M_k - \sigma_1^k \\
M_k &\nearrow M = S_0 \\
S_0 &\nearrow S_1 - \sigma_2^1 \\
&\vdots \\
S_{k-1} &\nearrow S_k - \sigma_2^k \\
S_k &\nearrow \Delta^n
\end{aligned}$$

If $\emptyset$ is an evader of A, the theorem holds if $\sigma_1^1 = \emptyset$ and $M_0 = M_1 = \emptyset$. This requires $M_2 = \sigma_1^2$.

Proof. First suppose that $\emptyset$ is not an evader of A. We will construct a discrete Morse function on Δ^n whose critical simplices are the evaders of A and induced gradient vector field $V_A = V$ minus the evader pairs. Let $W \subseteq V$ be the set of pairs $(\alpha, \beta) \in V$ such that either $(\alpha, \beta) \in M$ or $(\alpha, \beta) \notin M$. By Lemma 6.23, W is a gradient vector field on Δ^n. We now determine the critical simplices of W. By construction, a pair (α, β) of simplices of V is not in W if and only if $\alpha \in M$ and $\beta \notin M$ or $\alpha \notin M$ and $\beta \in M$; i.e., all evaders of A are critical simplices of W. Furthermore, the vertex v, which is paired with $\emptyset$, is also critical. This vertex and the

evaders of A form all the critical simplices of W. It remains to ensure that the ordering of expansions and attachments given in the statement of the theorem is respected.

To that end, let $f\colon \Delta^n \to \mathbb{R}$ be any discrete Morse function with induced gradient vector field W. By definition, if $\alpha^{(p)} \in M$ and $\gamma^{(p+1)} \notin M$ with $\alpha < \gamma$, then $(\alpha, \gamma) \notin W$ so that $f(\gamma) > f(\alpha)$. Define

$$\begin{aligned} a &:= \sup_{\alpha \in M} f(\alpha), \\ b &:= \inf_{\alpha \notin M} f(\alpha), \\ c &:= 1 + a - b, \\ d &:= \inf_{\alpha \in \Delta^n} f(\alpha), \end{aligned}$$

and a new discrete Morse function

$$g\colon \Delta^n \to \mathbb{R}$$

by

$$g(\alpha) := \begin{cases} f(\alpha) & \text{if } \alpha \in M - v, \\ f(\alpha) + c & \text{if } \alpha \notin M, \\ d - 1 & \text{if } \alpha = v. \end{cases}$$

To see that g is a discrete Morse function with the same critical simplices as f, observe that for each $\alpha \in M$ and $\beta \notin M$, $g(\beta) \geq c + 1 > c \geq g(\alpha)$. For each pair $\alpha^{(p)} < \beta^{(p+1)}$, we have $g(\beta) > g(\alpha)$ if and only if $f(\beta) > f(\alpha)$. Then g is a discrete Morse function with the same critical simplices as f.

The case where $\emptyset$ is an evader of A is similar. □

Exercise 6.29. Explain why the function f in Theorem 6.28 may not be the desired discrete Morse function. In other words, why was it necessary to construct the function g?

The proof of Theorem 6.21 follows immediately from Theorem 4.1; that is, for any decision tree A, $e_i(K) \geq b_i(K)$ where e_i is the number of pairs of evaders of index i. An immediate corollary is the following:

Corollary 6.30. For any decision tree algorithm A, the number of pairs of evaders of A is greater than or equal to $\sum_{i=0}^{n} b_i$.

Another corollary of Theorem 6.28 is the following:

Theorem 6.31. If M is nonevasive, then M is collapsible.

Problem 6.32. Prove Theorem 6.31.

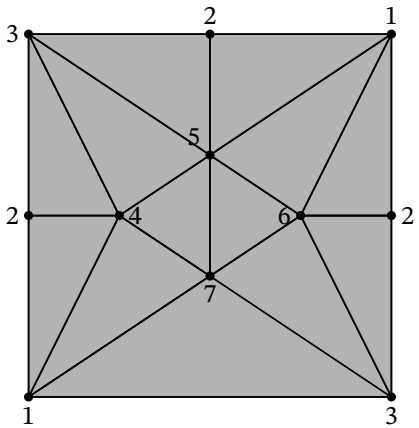
An interesting question is whether or not the converse of Theorem 6.31 is true. That is, is nonevasive the same thing as collapsible? The following example answers this question in the negative. One concept that will be helpful is that of the **link** of a vertex. We define it below and will study it in more detail in Chapter 10.

Definition 6.33. Let K be a simplicial complex and $v \in K$ a vertex. The **star of v in K**, denoted by $\mathrm{star}_K(v)$, is the simplicial complex induced by the set of all simplices of K containing v. The **link of v in K** is the set $\mathrm{link}_K(v) := \mathrm{star}_K(v) - \{\sigma \in K : v \in \sigma\}$.

The following lemma gives a sufficient condition for detecting evasiveness.

Lemma 6.34. If $M \subseteq \Delta^n$ is nonevasive, then there exists a vertex $v \in M$ such that $\mathrm{link}_{\Delta^n}(v)$ is nonevasive.

Example 6.35. Let C be the simplicial complex below.



Note that there is only one free pair (namely, $\{13, 137\}$) and that starting from this free pair, we may collapse C. Showing that C is evasive is Problem 6.36.

Problem 6.36. Show that C in Example 6.35 is evasive.