Computer Science B36 — Fall 2014
Scarborough Campus — University of Toronto

Sliding puzzle: An introduction to invariants

Vassos Hadzilacos

An $n \times n$ sliding puzzle is a matrix with n rows and n columns, whose entries contain, in some arbitrary order, the integers $1, 2, \ldots, n^2-1$ and a special value called "blank" and denoted $\perp$. A particular assignment of these n^2 values to the n^2 entries of the puzzle is called a ***configuration*** of the sliding puzzle. Shown below are two configurations of a 4×4 sliding puzzle.

11	2	9	4
5		7	8
3	10	6	12
13	15	14	1

(a)

1	2	14	
5	6	13	8
9	10	11	12
4	3	15	7

(b)

Figure 1: Two configurations of a 4×4 sliding puzzle ($\perp$ shown as a blank square)

We now define "moves" from a configuration C.

Definition 1 *Two entries in a sliding puzzle are* ***adjacent*** *if one is above, below, to the left, or the the right of the other. More precisely, (i, j) and (i', j') are adjacent if and only if one of the following conditions applies:*

- *$i > 1$, $i' = i - 1$, and $j' = j$ (so, (i', j') is above (i, j)); or*
- *$i < n$, $i' = i + 1$, and $j' = j$ (so, (i', j') is below (i, j)); or*
- *$i' = i$, $j > 1$, and $j' = j - 1$ (so (i', j') is to the left of (i, j)); or*
- *$i' = i$, $j < n$, and $j' = j + 1$ (so (i', j') is to the right of (i, j)).*

Definition 2 *Let C and C' be configurations of an $n \times n$ sliding puzzle. We can* ***transform C to C' in one move*** *iff C' is obtained from C by swapping the blank entry in C with the integer in any one of the adjacent entries; all entries except the two that were swapped are the same in C' as in C.*

The preceding is a mathematical description of a well-known game consisting of an $n \times n$ frame divided into n^2 slots of equal size, one of which is empty and the remaining contain tiles labeled with the numbers $1, 2, \ldots, n^2 - 1$. We can transform one arrangement of the tiles within the frame (i.e., a configuration) to another in a single move by (and only by) sliding to the empty slot one of the tiles adjacent to it, thereby making empty the slot previously occupied by the tile we moved. Given the frame with the tiles in some scrambled order, the objective of the game is to apply a sequence of sliding moves of the kind described above and transform it so that the tiles are in the "natural" order: 1 through n in the first row (in left to right order), $n+1$ through $2n$ in the second row (left to right), and so on, with row n containing tiles n^2-n+1 through n^2-1 (left to right) in the first $n-1$ columns, and the bottom right slot being empty. Sometimes the tiles don't have numbers on them, but fragments of a picture that is formed if the tiles are placed in one specific order. We can think of that order as the "natural" one, and the tiles as being labeled with numbers instead of picture fragments.

Let C_n^* be the configuration of an $n \times n$ sliding puzzle in which the matrix entries, listed left to right from row 1 to row n, in that order, contain $1, 2, \ldots, n^2-1, \perp$ (i.e., the tiles are in their "natural" positions). Figure 2(b) shows C_3^*. Our general problem is the following: Given a configuration C of an $n \times n$ sliding puzzle, is it possible to transform C to C_n^* in a finite sequence of moves?

A particular instance of this problem is shown below: Is there a sequence of moves that transforms the configuration in Figure 2(a) to the configuration in Figure 2(b)?

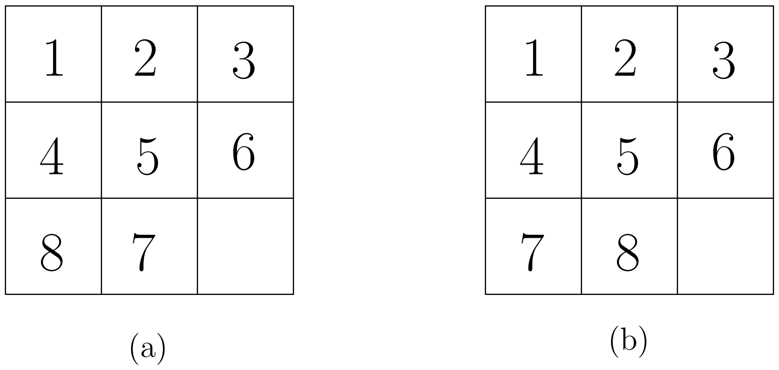

Figure 2: Two configurations of a 3×3 sliding puzzle

Prima facie, this appears to be an easy task, since the starting configuration is very close the the target configuration: just two slots are out of order. After trying it for a while, however (do it!), you will discover that it is not so easy after all. You may even start suspecting that it cannot be done at all. Indeed, as we will see, this is the case.

How can we prove that this task is impossible? Convincing someone (i.e., proving) that we ***can*** transform an initial configuration to a target configuration is, at least methodologically, straightforward: We simply demonstrate a sequence of legal moves that does the job. But how can we prove that no such sequence of moves exists? We can't try all possible sequences of moves, because the sequences are arbitrarily long and so there is an infinite number of them. We can be a bit more clever and create a directed graph of configurations (of which there are only finitely many), with an edge from configuration C to configuration C' if and only if we can transform C to C' in one move. We would then discover that there is no path leading from the initial configuration (the one in Figure 2(a)) to the target configuration (the one in Figure 2(b)), and therefore there is no sequence of moves to do the job. But this would be a monumental undertaking, as there are $9! = 362,880$ configurations to deal with. Furthermore, this argument wouldn't be very convincing to another person, who would have to verify that the graph we constructed (or the program we wrote to construct the graph) is correct! Fortunately, a much more elegant argument proves that it is impossible to transform the initial configuration to the target configuration. This argument also illustrates the important concept of ***invariants***.

The general idea is simple: We will find a property that holds in all configurations reachable from the initial one, after any number of moves (such a property is called an "invariant"), but which the target configuration in does ***not*** have. This implies that the target configuration cannot be reached from the initial one, after any number of moves.

We can define a total order $<$ on the slots of an $n \times n$ matrix as follows: $(i,j) < (i',j')$ iff $i < i'$ or $i = i'$ and $j < j'$. In other words, if we list the slots left to right, starting from row 1 and ending with row n, in that order, we encounter (i,j) before (i',j').

Definition 3 *An **inversion** of a configuration C is a pair of slots (i,j) and (i',j') such that $(i,j) < (i',j')$ but in configuration C slot (i,j) contains a larger integer than slot (i',j').*

Intuitively, an inversion is a pair of slots of C that contain integers that are "out of order" relative to their order in the target configuration C_n^*.

Definition 4 *A **horizontal** move is a move that involves swapping the blank entry of a configuration with the entry to its left or its right; a **vertical** move is a move that involves swapping the blank entry of a configuration with the entry above or below it.*

Lemma 5 *Let C be a configuration of a 3×3 sliding puzzle and C' be a configuration that results from C after one horizontal move. Then the number of inversions in C' is the same as the number of inversions in C.*

PROOF. In a horizontal move, we swap the contents of a slot that contains an integer with the adjacent slot to its left or right that contains $\perp$. Thus, the relative order of all the integers listed in slot order (recall the total order on the slots defined earlier), is the same in C as in C'. Thus, C and C' have the same number of inversions. $\square$

Lemma 6 *Let C be a configuration of a 3×3 sliding puzzle and C' be a configuration that results from C after one vertical move. Then the number of inversions in C' is the same as, two more than, or two less than the number of inversions in C.*

PROOF. In a vertical move, we swap the contents of a slot that contains an integer X with the adjacent slot above or below that contains $\perp$. In a 3×3 puzzle this means that, in C', X "skips over" exactly the two slots immediately after or immediately before it in the total ordering of slots, and occupies the slot that was blank in C, and the blank slot in C' is where X was in C. Let Y and Z be the integers in the slots that X "skipped over". Thus, the only difference in the inversions of C and C' are the inversions involving X, Y, and Z: all other integers are in the same relative order in C and C' with respect to each other and with respect to X, Y, and Z. There are three cases, depending on the relative order of X with respect to Y and Z.

CASE 1. X is less than both Y and Z. Then the number of inversions in C' is two more than the number of inversions in C, if X moved down (i.e., the slots that contain Y and Z are immediately after the slot that contains X in C); and it is two fewer than the number of inversions in C, if X moved up (i.e., the slots that contain Y and Z are immediately before the slot that contains X in C).

CASE 2. X is less than one of Y and Z and larger than the other. Then the number of inversions in C' is the same as the number of inversions in C.

CASE 3. X is greater than both Y and Z. Then the number of inversions in C' is two less than the number of inversions in C, if X moved down; and it is two more than the number of inversions in C, if X moved up.

□

Corollary 7 *Let C be a configuration of a 3×3 sliding puzzle and C' be a configuration that results from C after one move. Then the number of inversions of C and C' have the same parity (i.e., are both odd or both even).*

Lemma 8 *Let C be the configuration of a 3×3 sliding puzzle shown in Figure 2(a). Every configuration that results from C after any number of moves has an odd number of inversions.*

Proof. Let $P(n)$ be the following predicate: Every configuration that results from C after n moves has an odd number of inversions. To prove the lemma it suffices to show that $P(n)$ is true for every $n \in \mathbb{N}$. We do so using induction.

Basis: $n = 0$. The only configuration that results from C after zero moves is C itself, which has an odd number of inversions. Thus, $P(0)$ holds.

Induction Step: Let i be an arbitrary natural number, and suppose that $P(i)$ holds. Let C' be any configuration that results from C after $i + 1$ moves. Thus, there is a sequence of configurations $C_0, C_1, C_2, \ldots, C_i, C_{i+1}$ so that $C_0 = C$, $C_{i+1} = C'$, and, for every j such that $0 \leq j \leq i$, C_j can be transformed to C_{j+1} in one move. Therefore, C_i results from C after i moves. By the induction hypothesis, C_i has an odd number of inversions. By Corollary 7, the number of inversions in C_{i+1} and in C_i have the same parity. Thus, C_{i+1} has an odd number of inversions. We have shown that an arbitrary configuration C' that results from C in $i + 1$ moves has an odd number of inversions. So, $P(i + 1)$ holds.

Thus, $P(n)$ holds for every natural number n. □

Theorem 9 *It is not possible to transform the configuration in Figure 2(a) to the configuration in Figure 2(b) in any number of moves.*

Proof. The configuration in Figure 2(b) has zero inversions, i.e., an even number of inversions. The theorem now follows by Lemma 8. □

The preceding proof is an example of the use of invariants to prove properties of "discrete-event systems". A discrete-event system is a system whose behaviour evolves as a sequence of states: it starts in a particular state s_0; then, as a result of the occurrence of some event, it transitions to a new state s_1; then, as a result of the occurrence of another event, it transitions to a new state s_2, and so on. For example, a beverage dispensing machine can be described as a discrete-event system: It starts in some initial state, with a specific number of beverages of specific types — say 10 bottles of orange juice, 15 bottles of water, and 15 cans of soda — and a specific number of coins of specific types to make change — say, 25 toonies, 25 loonies, 50 quarters, 100 dimes, and 70 nickels. Then, as a result of an event — say a customer depositing 2 toonies and pressing the button for a bottle of orange juice that costs 3.60 — the machine dispenses the requested beverage and enters a new state — where it has 9 bottles of orange juice, 15 bottles of water, and 15 cans of soda, 27 toonies, 25 loonies, 49 quarters, 99 dimes, and 69 nickels. Another example of a discrete event system is the execution of a computer program. It starts in a specific state, with its variables in their initial states, and the program counter indicating the location of the first instruction of the program; after the execution of the first instruction it enters a new state, with its variables modified depending on the executed instruction, and the program counter indicating the location of the second instruction of the program; and proceeds similarly entering new states after the execution of each instruction. In the example of a sliding puzzle, the initial state is the starting configuration of the puzzle; and a new state is reached by performing a move (i.e., sliding one of the tiles adjacent to the empty slot to the position of that slot).

An ***invariant*** of a discrete event system is a predicate on the states with the following property: It is true in the initial state of the system, and is ***preserved*** by every state transition of the system. That is, if it is true in state s it is also true in every state s' of the system that can be reached from s by the execution of one event of the system. Therefore, by the principle of induction, an invariant is true in every state that can be reached from the initial state of the system.

Given a description of a discrete-event system (for example, a program controlling an aircraft in flight) we want to prove that the system never enters a state s in which an undesirable property $P(s)$ holds (for example, at take-off the front wheels are off the ground but the engine thrust is below a certain threshold). One way of proving the correctness of such a system is to prove that some predicate $Q(s)$, which implies that $P(s)$ does ***not*** hold, is an invariant of the system. This shows that the system is always in a state where Q holds, and therefore (since Q implies that P does not hold) that the system is always in a state in which the the undesirable property P does not hold.

Let us now review how we applied this methodology to our sliding puzzle example. Here, our goal is to prove that we cannot transform the configuration in Figure 2(a) to the configuration in Figure 2(b). So, the "undesirable" state is the configuration in Figure 2(b). The property $Q(s)$ (where the state s is a configuration of the 3×3 sliding puzzle) is that the number of inversions in s is odd. Clearly this implies that s is not the configuration in Figure 2(b) (because that configuration has zero inversions, an even number). In Lemma 8, we proved that Q is an invariant of the system. Therefore the configuration in Figure 2(b) cannot be reached starting from the configuration in Figure 2(a).