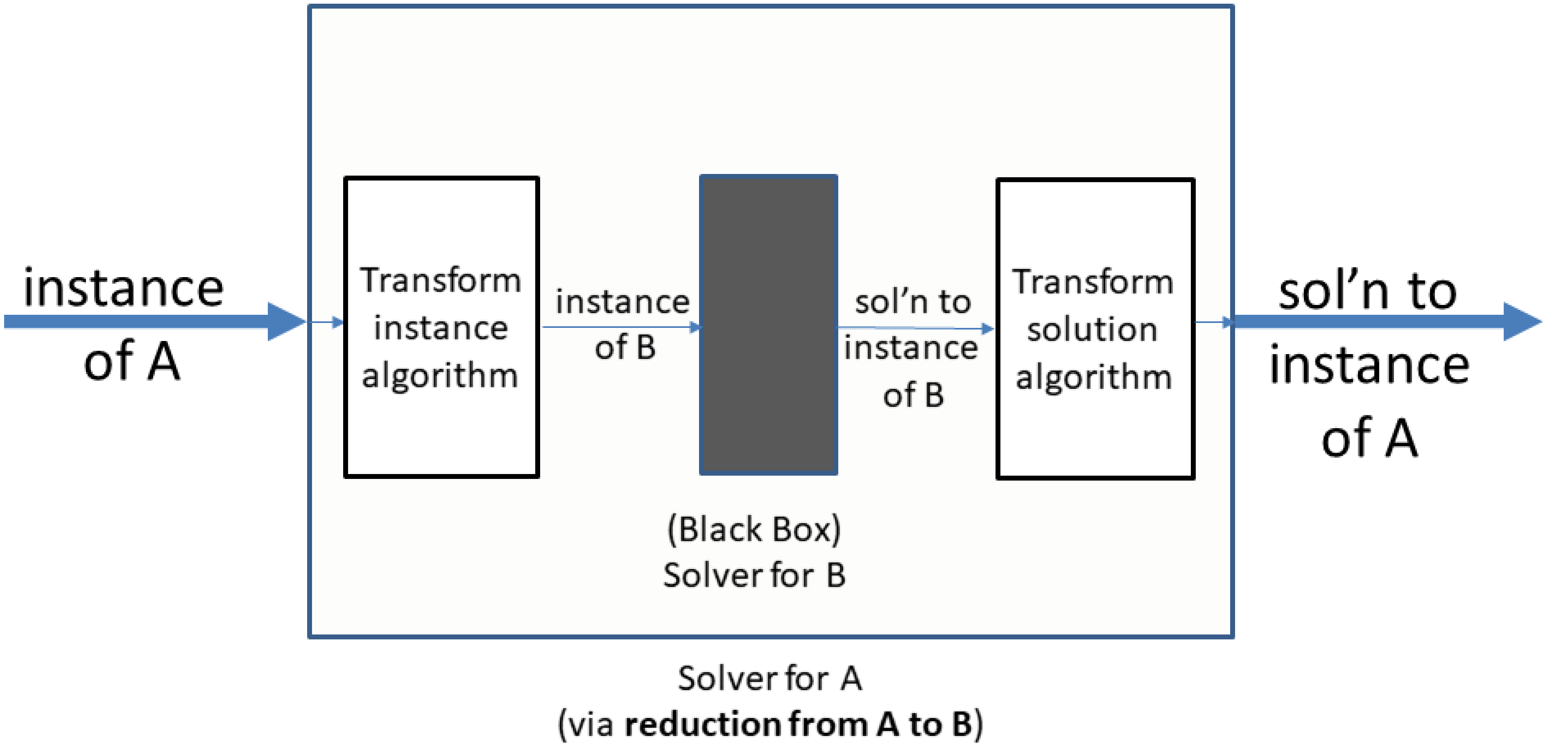
2. Reductions

Context

- You encounter a new problem A
  - You don't have an algorithm for it.
  - But you can transform it into another problem B for which you have an algorithm.
  - So you don't really need to design an algorithm to solve problem A.
- Definition: An *instance* of a problem is a valid input, drawn from the space of inputs the problem allows.

How a reduction works



How a reduction works (continued)

- You need to:
  - show how to transform an arbitrary instance I_A of A into an instance I_B of B.
  - show how to transform the solution S_B of I_B into a solution S_A of I_A.
  - prove that S_A is a correct solution for I_A.
- The total running time is
  - The sum of the times of the two transformations
  - plus the time to solve the instance I_B.

Reduction example

- A: Given a set $\{ x_1, x_2, \ldots, x_n \}$ of integers, find the smallest gap between any two of them.
  - That is, find $\min_{i, j \in \{1, \ldots, n\}} \{ |x_i - x_j| \}$
- B: sorting a list of values
- Reduction:
  - given an instance I_A of A, let $I_B = I_A$.
  - sort the list I_B to get a list $\{ y_1, y_2, \ldots, y_n \}$.
  - return $\min_{i = 1, 2, \ldots, n-1} \{ y_{i+1} - y_i \}$.