Constraint Systems

Restarts and LNS in or-tools

Restarts in or-tools

It is very easy to use restarts in or-tools

They are handled via a search monitor:

- The monitors tracks the number of fails…
- …And causes a restart whenever the limit is reached

There are two types of restart search monitor:

```
slv.LubyRestart(scale) # Luby restart sequence
slv.ConstantRestart(limit) # Constant limit
```

We will also need to randomize the search strategy:

- We can use `CHOOSE_RANDOM_VAR` for variable selection…
- …Or `ASSIGN_RANDOM_VALUE` for value selection

LNS in or-tools

Or-tools allows to specify easily a fragment selection strategy

We just need to implement a simple python class:

```
class MyRandomLns(pywrapcp.BaseLns):
  def __init__(self, x, size, rand):
    pywrapcp.BaseLns.__init__(self, x)
    self.size_ = size # Number of vars to relax (e.g.)
    self.rand_ = rand # RNG

  def InitFragments(self):
    pass

  def NextFragment(self):
    positions =
    for pos in positions:
      self.AppendToFragment(pos)
    return True
```

- `__init__` is called when the class is built

LNS in or-tools

Or-tools allows to specify easily a fragment selection strategy

We just need to implement a simple python class:

```
class MyRandomLns(pywrapcp.BaseLns):
  def __init__(self, x, size, rand):
    pywrapcp.BaseLns.__init__(self, x)
    self.size_ = size # Number of vars to relax (e.g.)
    self.rand_ = rand # RNG

  def InitFragments(self):
    pass

  def NextFragment(self):
    positions = # random indices of the variables
    for pos in positions:
      self.AppendToFragment(pos)
    return True
```

- `NextFragment` specifies the variables to relax

LNS in or-tools

Or-tools allows to specify easily a fragment selection strategy

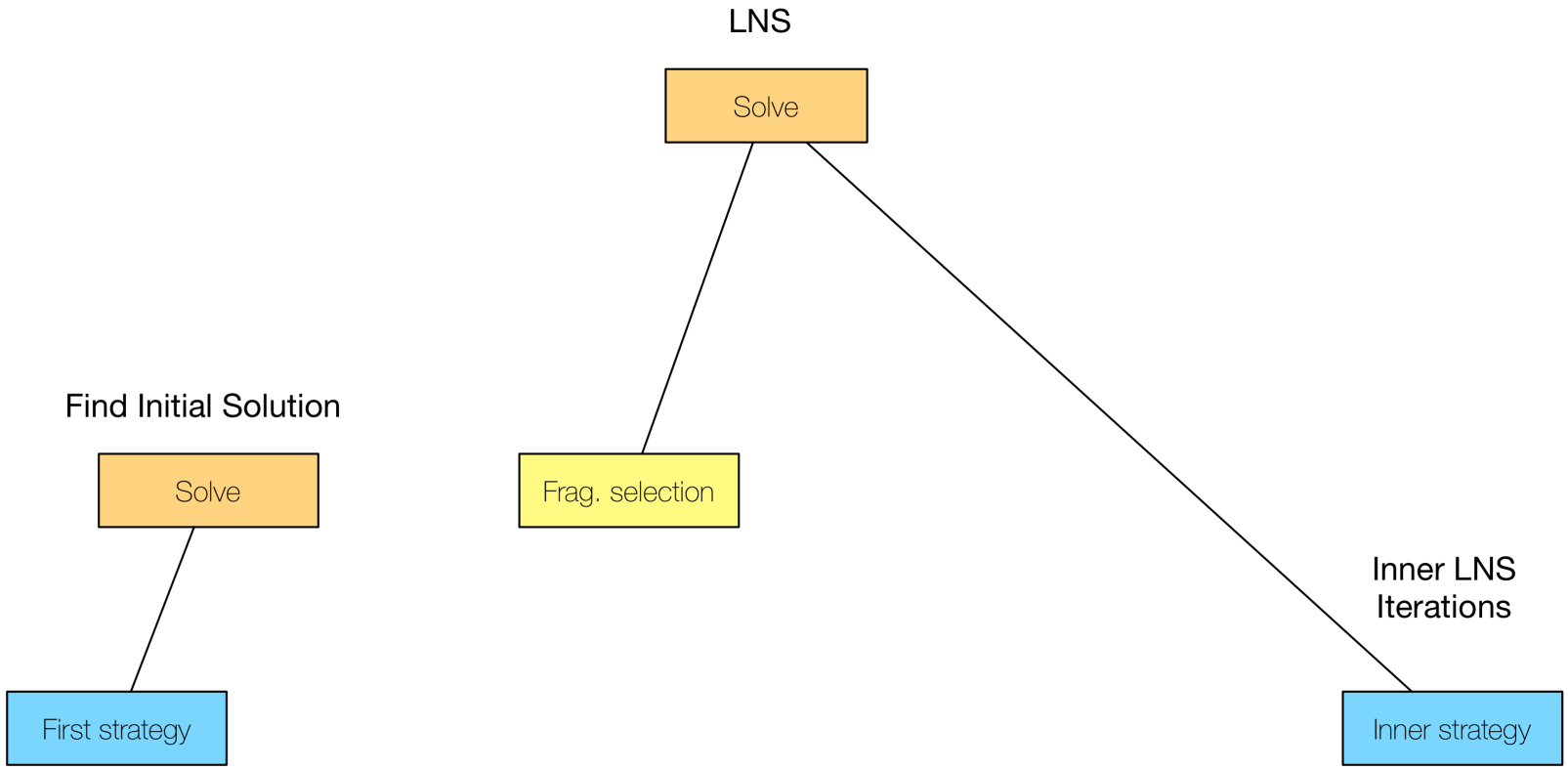
We just need to implement a simple python class:

```
class MyRandomLns(pywrapcp.BaseLns):
  def __init__(self, x, size, rand):
    pywrapcp.BaseLns.__init__(self, x)
    self.size_ = size # Number of vars to relax (e.g.)
    self.rand_ = rand # RNG

  def InitFragments(self):
    pass

  def NextFragment(self):
    positions = # random indices of the variables
    for pos in positions:
      self.AppendToFragment(pos)
    return True
```

- `InitFragments` is called whenever an improving solution is found

LNS in or-tools

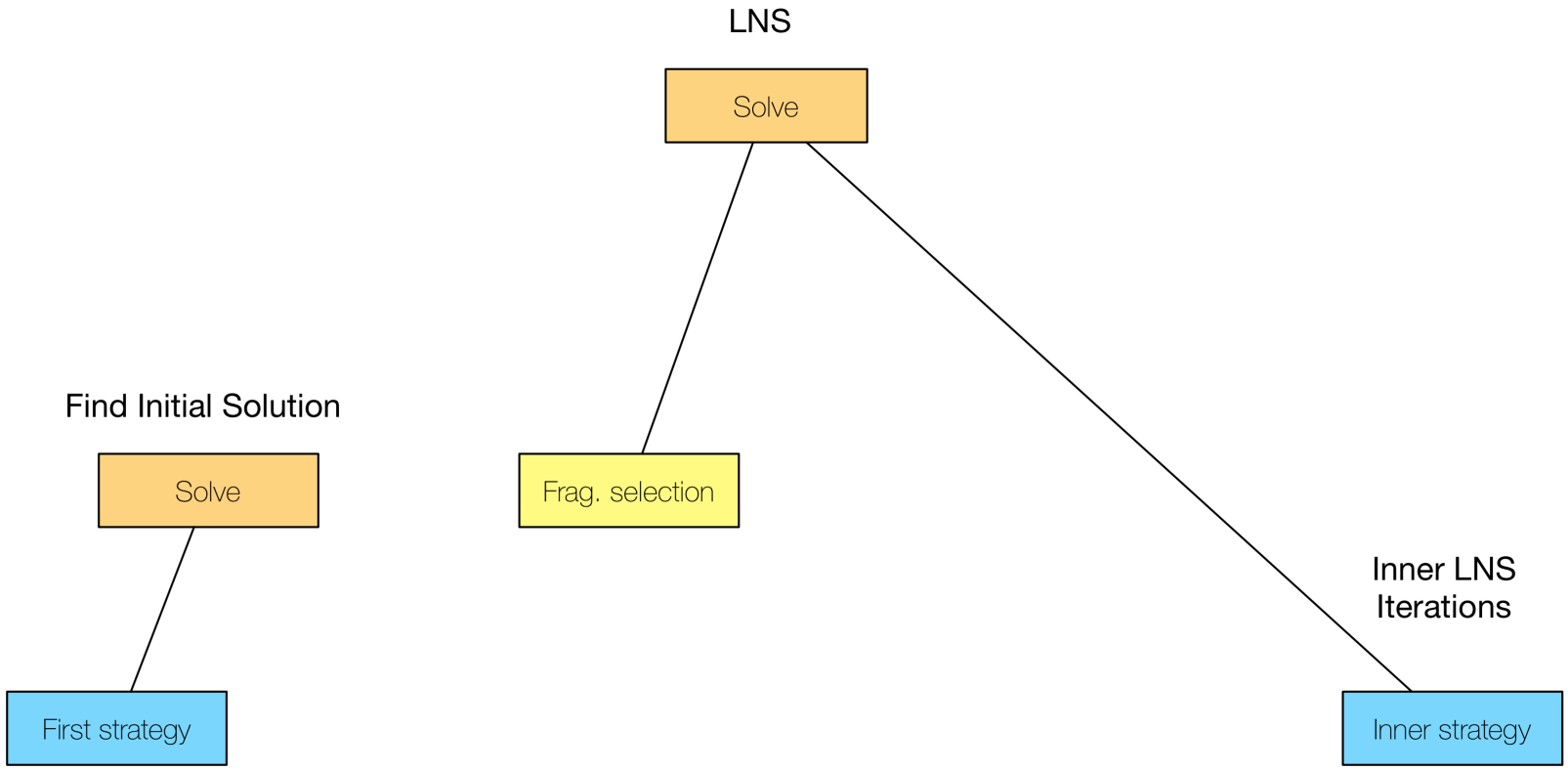
The tricky part is setting up the LNS process



- Key elements: initial search, fragment selection, inner search

LNS in or-tools

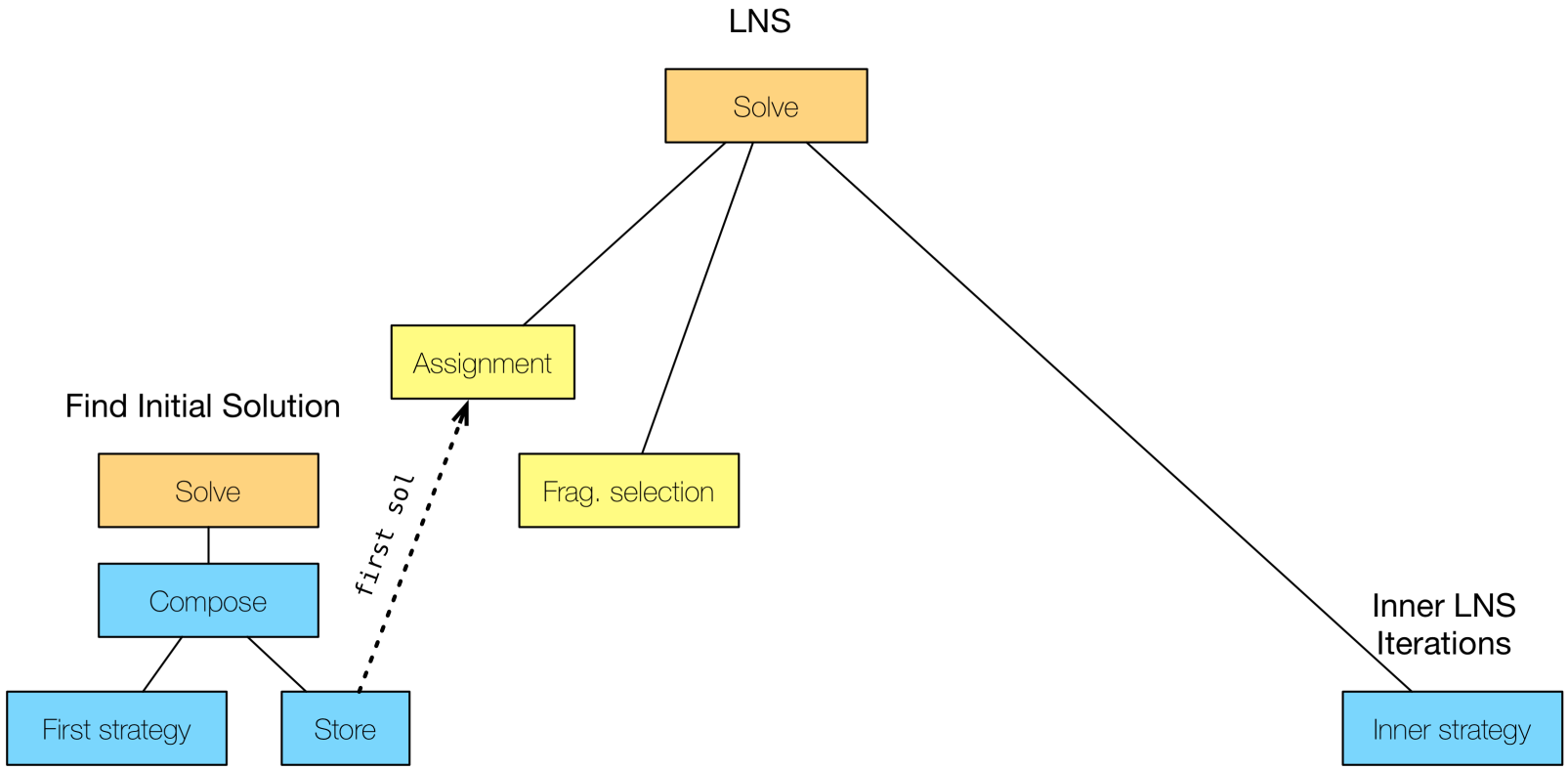
The tricky part is setting up the LNS process



- Blue: decision builders, orange: calls to **`Solve`**, yellow: other

LNS in or-tools

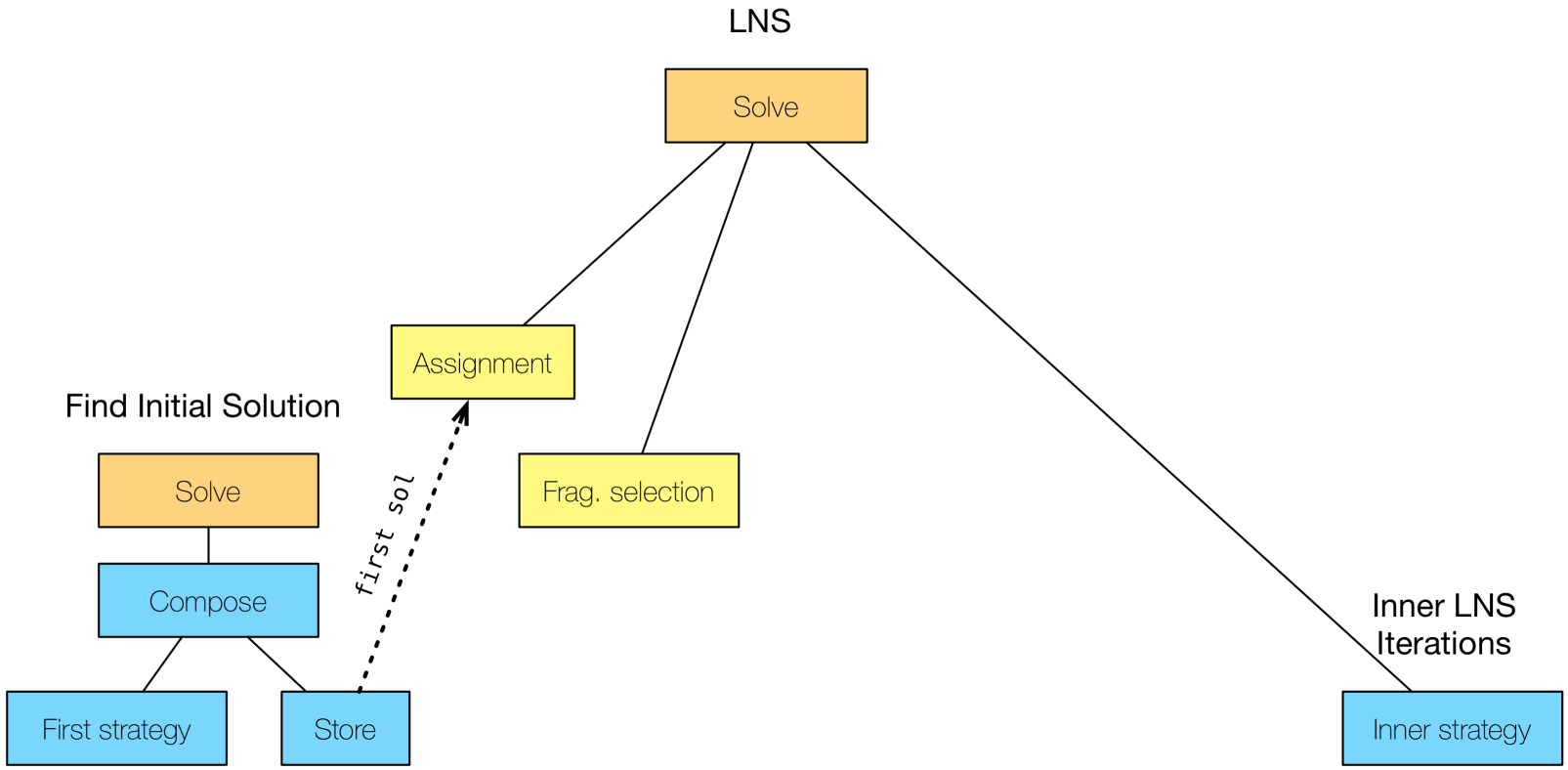
The tricky part is setting up the LNS process



- But we need to pass the initial solution to the LNS process…

LNS in or-tools

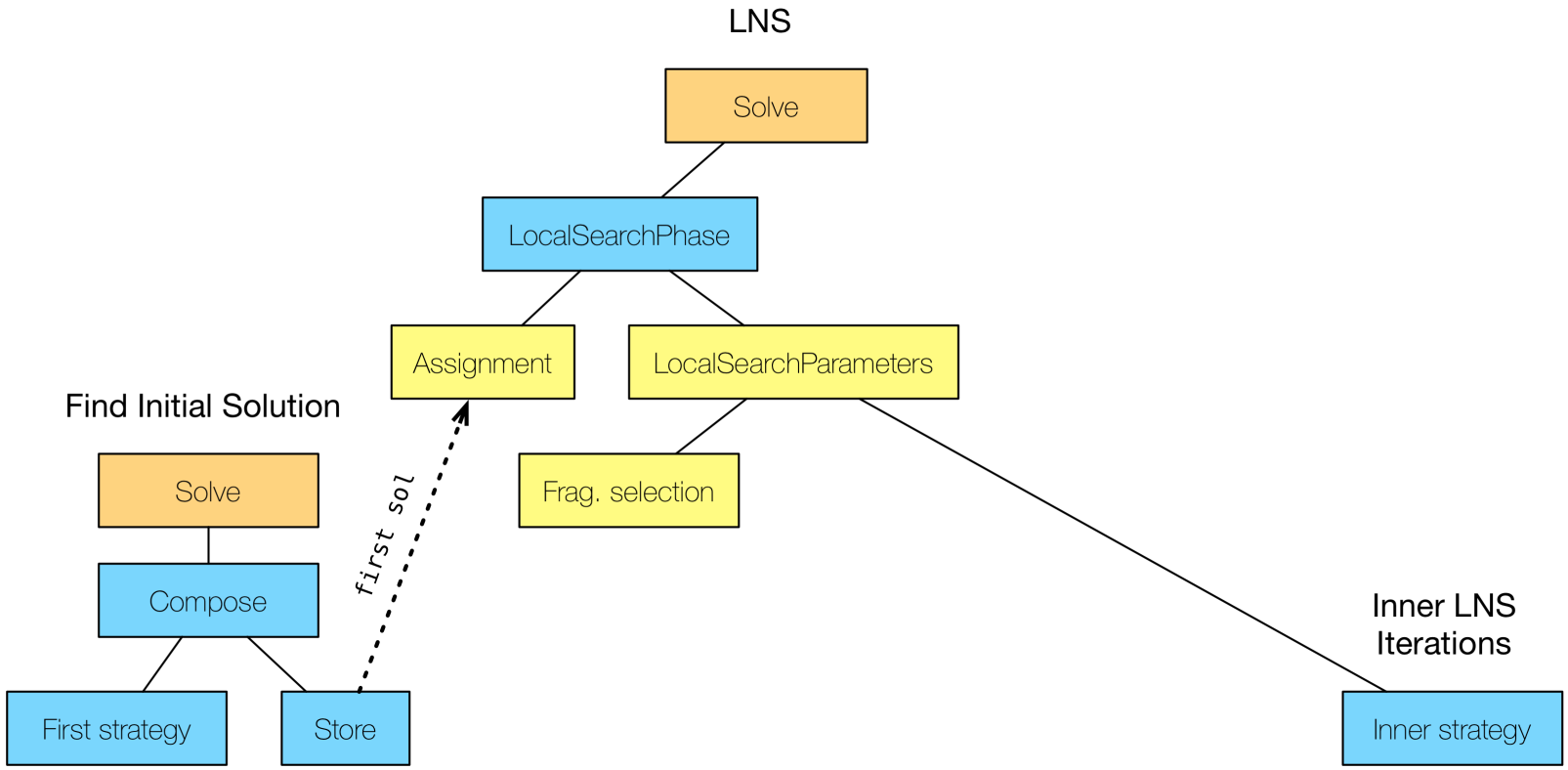
The tricky part is setting up the LNS process



- ...We use an **`Assignment`** object, stored by a special dec. builder

LNS in or-tools

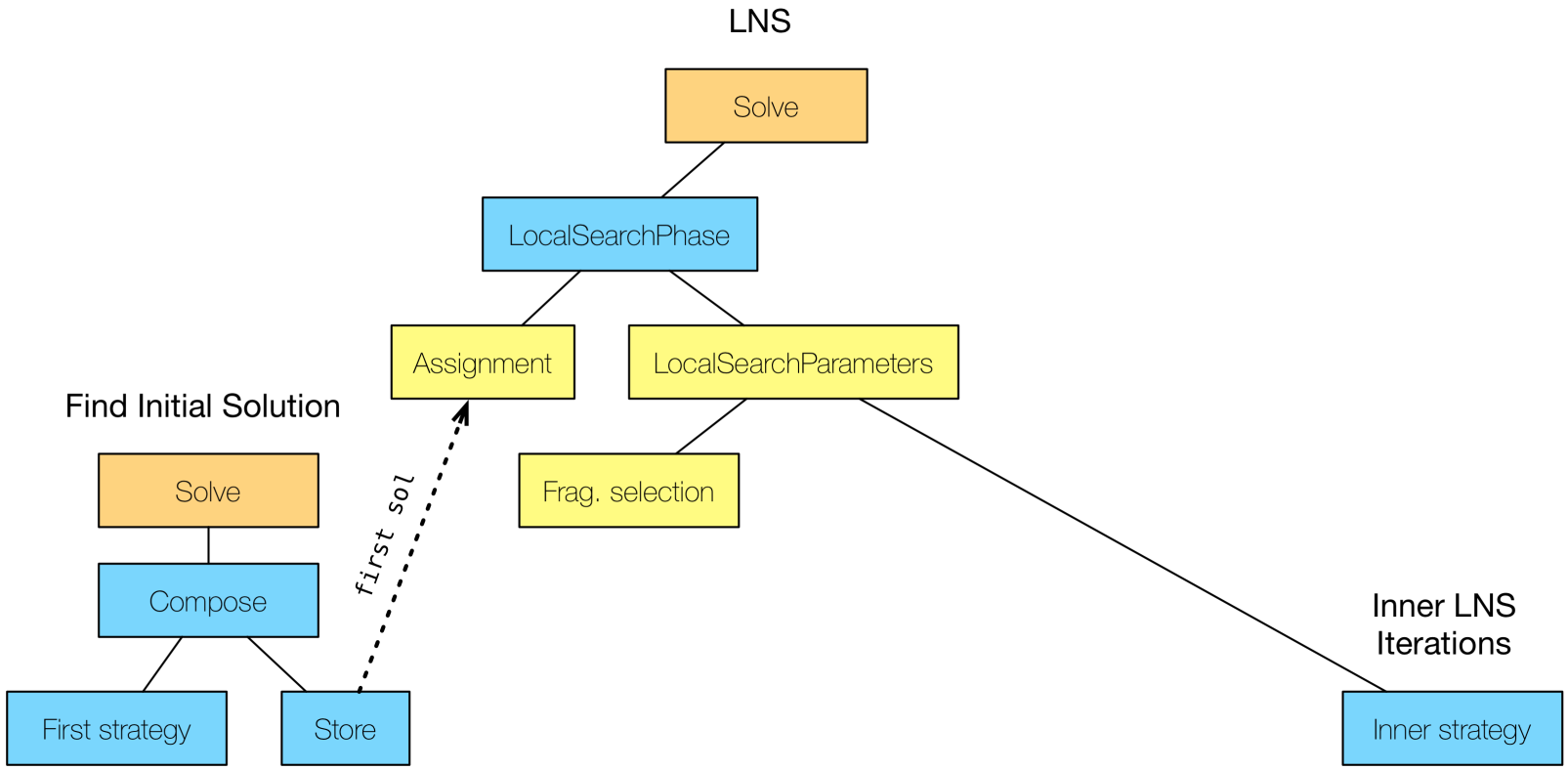
The tricky part is setting up the LNS process



- `Solve` needs a `DecisionBuilder` as argument...

LNS in or-tools

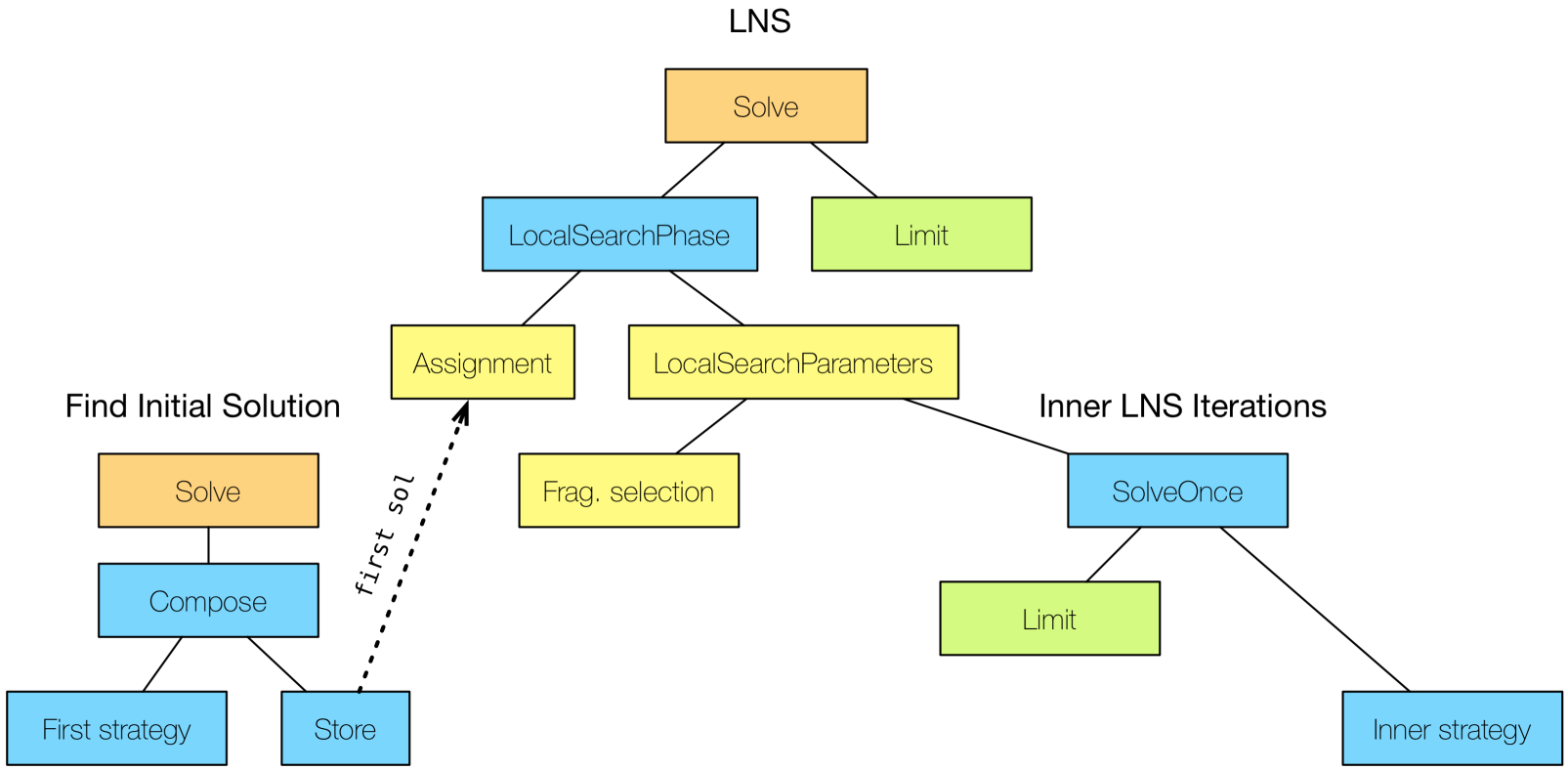
The tricky part is setting up the LNS process



- ...So we need a **`DecisionBuilder`** to control the LNS process

LNS in or-tools

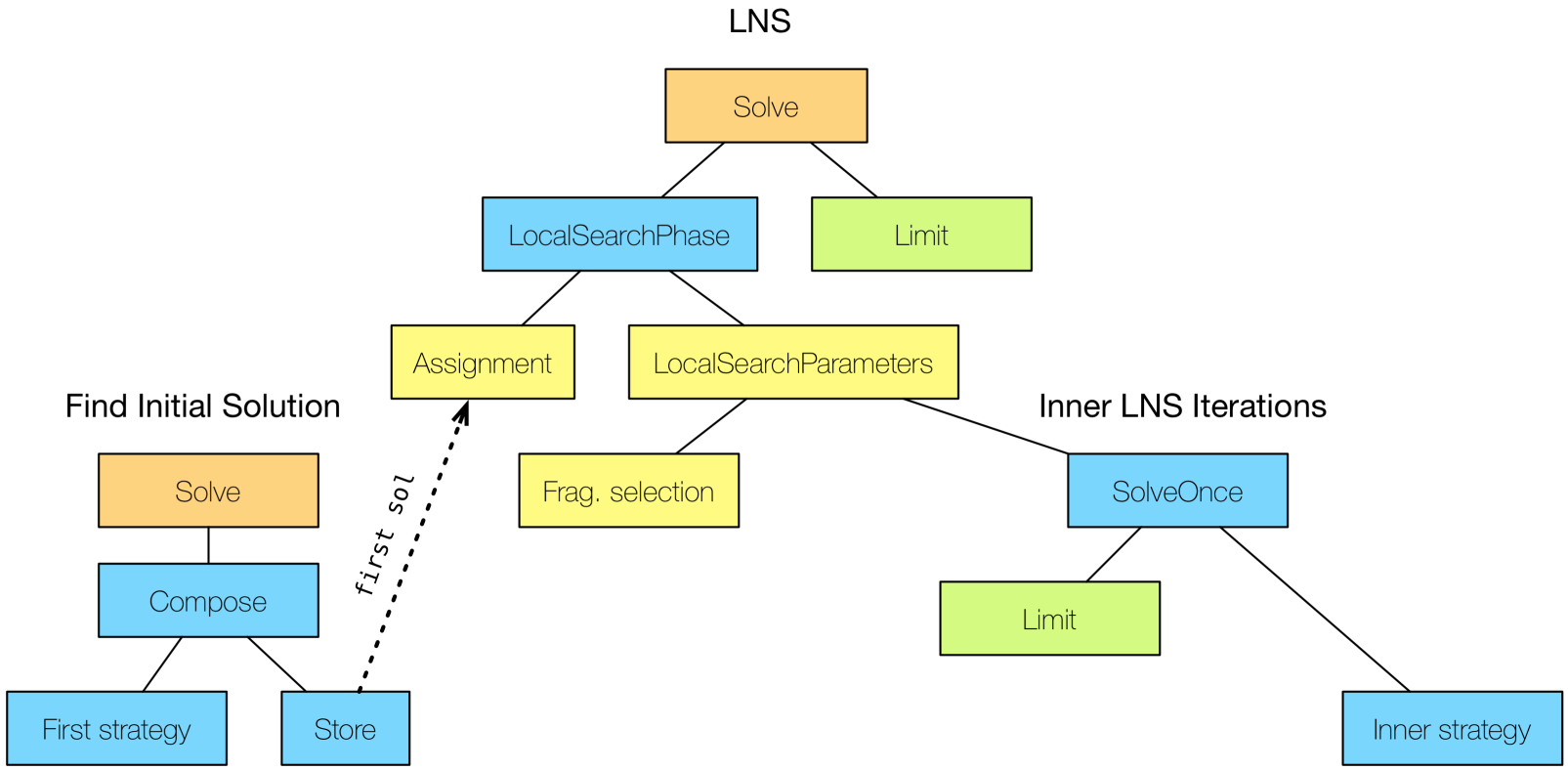
The tricky part is setting up the LNS process



- We will want to specify a global limit on the LNS phase...

LNS in or-tools

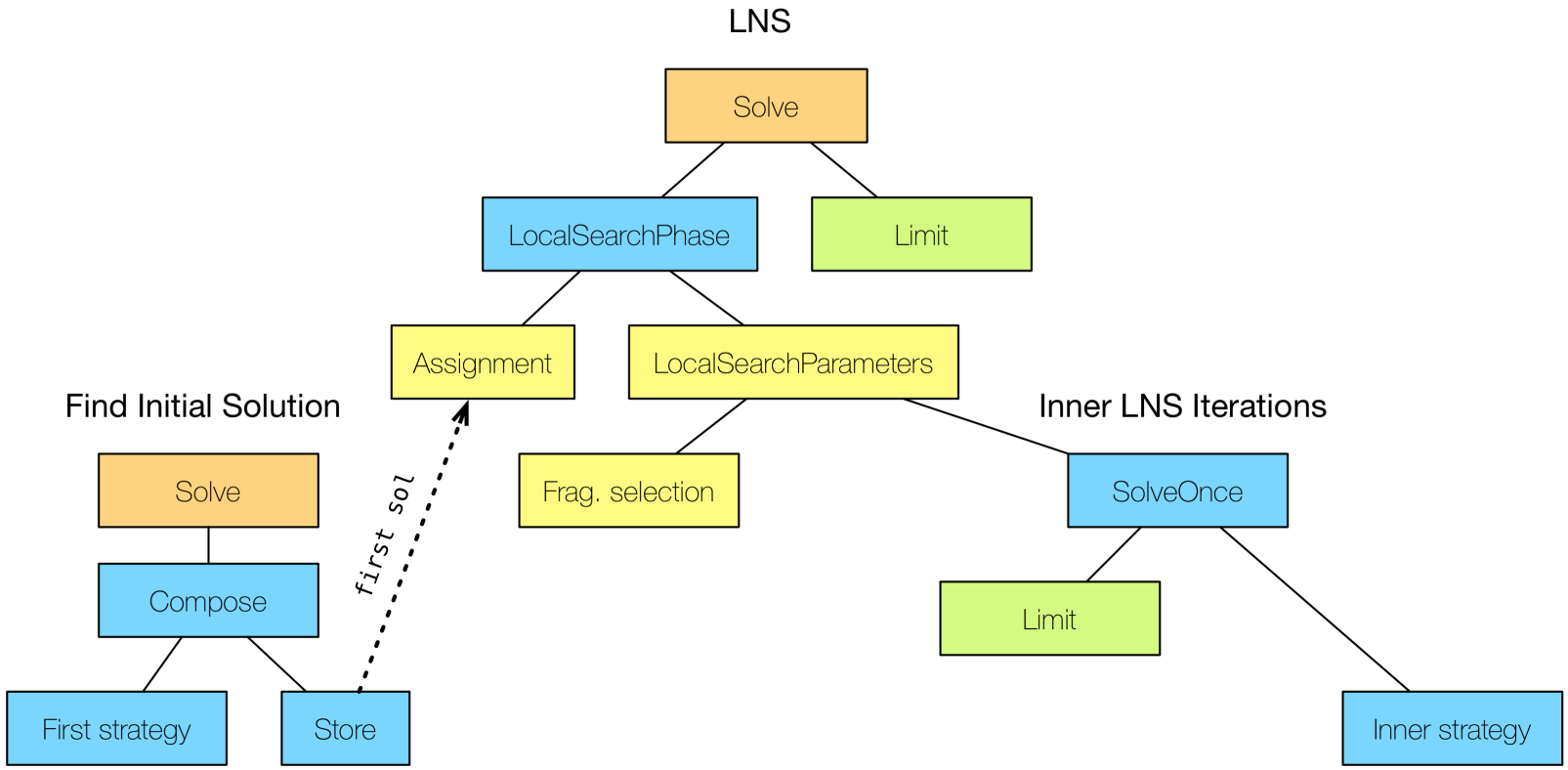
The tricky part is setting up the LNS process



- ...And a different local limit for each LNS iteration

LNS in or-tools

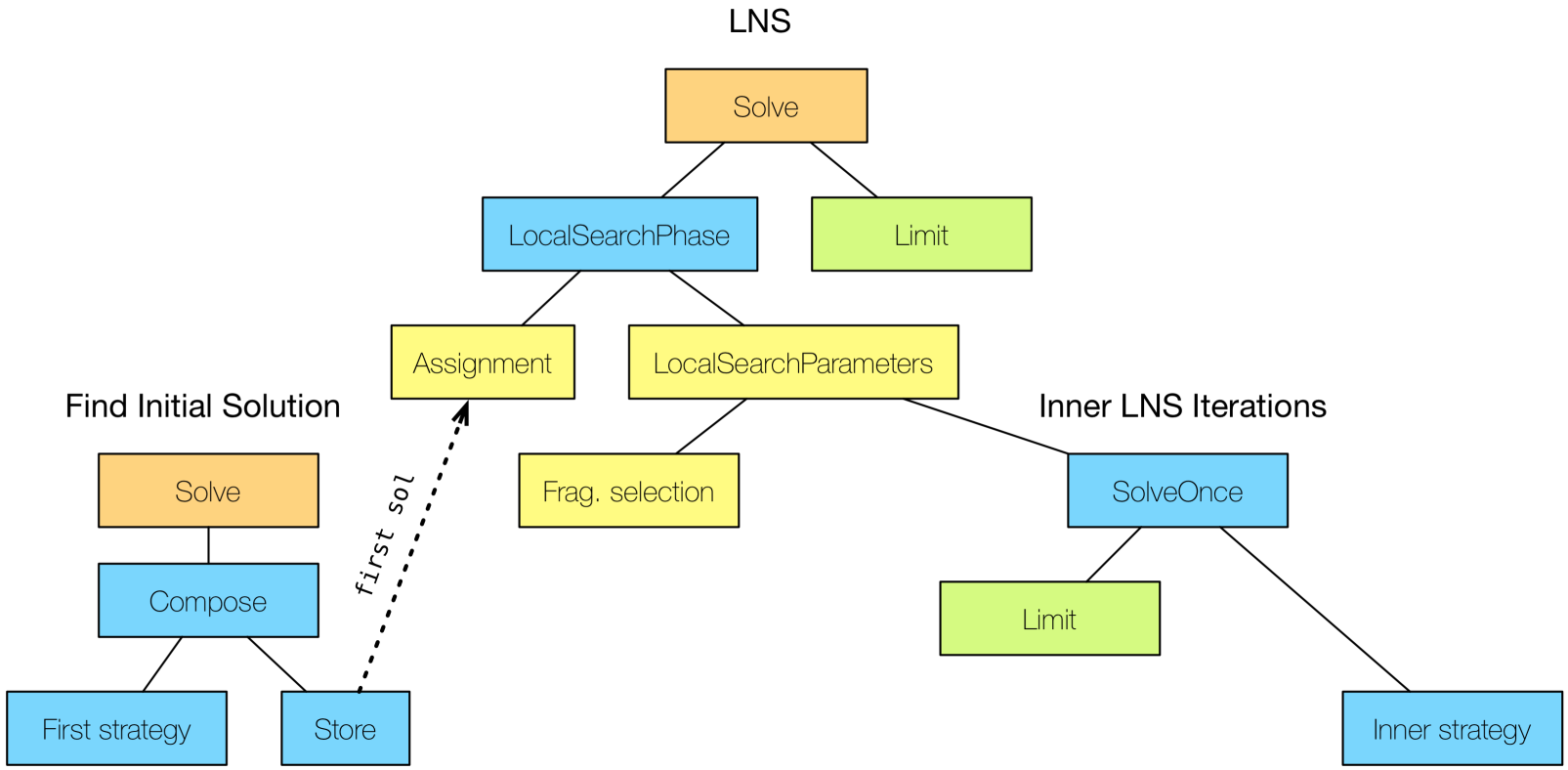
The tricky part is setting up the LNS process



- We do it via a special **`SolveOnce`** decision builder...

LNS in or-tools

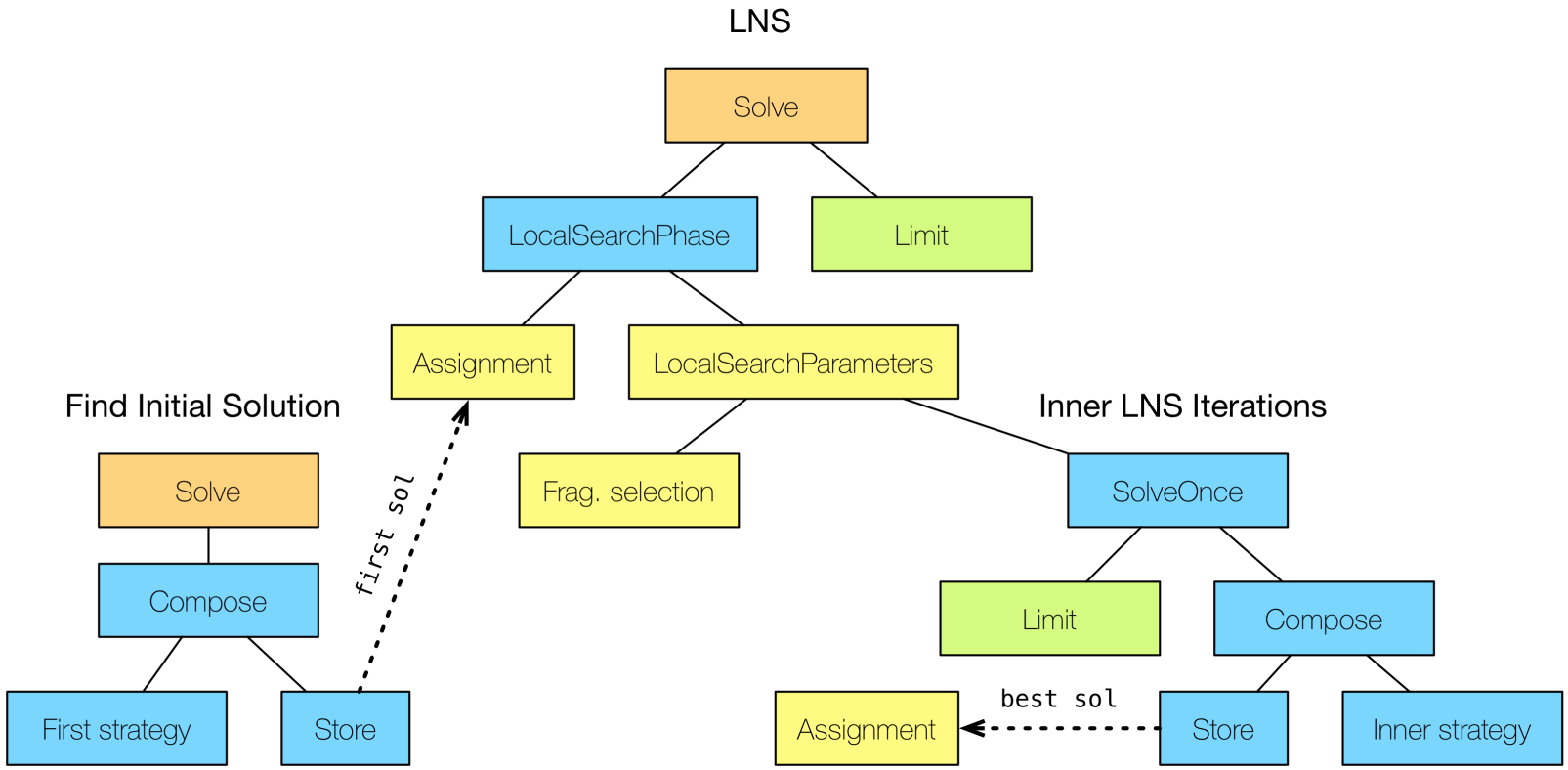
The tricky part is setting up the LNS process



- ...That can apply monitors to a sub-search

LNS in or-tools

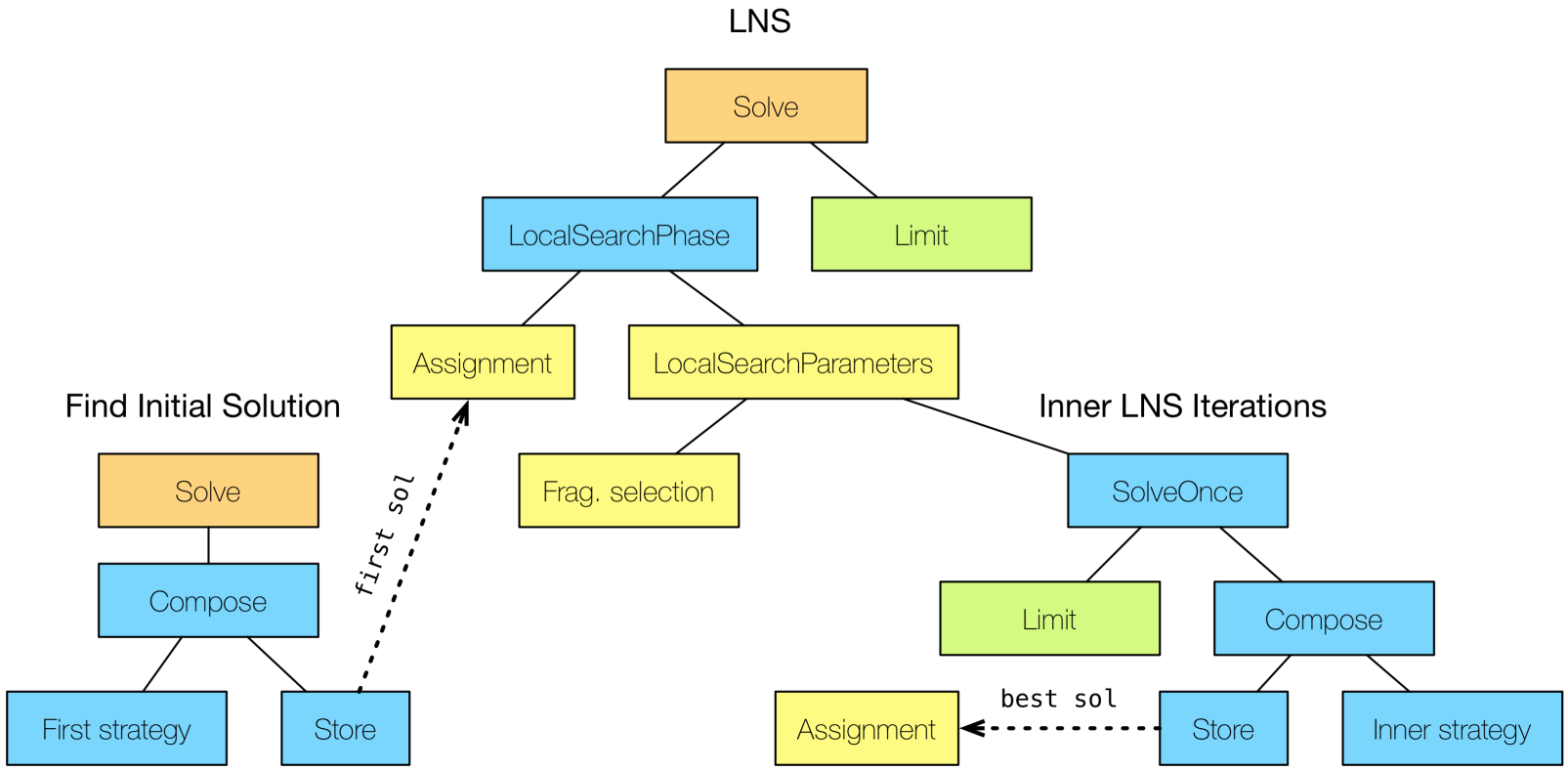
The tricky part is setting up the LNS process



- Finally, we may want to store the best sol via an **`Assignment`**

LNS in or-tools

The tricky part is setting up the LNS process



- This is not stricly needed, but it may help reusing some code

Constraint Systems

Restarts and LNS
for Shift-Scheduling

Restarts and LNS for Shift-Scheduling

Let's apply these techniques to our shift-scheduling problem

In the start kit, you will find two templates:

- **`shiftsched_restart.py`**, to test a restart strategy
- **`shiftsched_lns.py`**, to test a LNS

For the "restart" case:

- Try to add a restarting monitor
- Test different ways to randomize search (vars/values)
- Play with different values for the limit

Restarts and LNS for Shift-Scheduling

Let's apply these techniques to our shift-scheduling problem

In the start kit, you will find two templates:

- **`shiftsched_restart.py`**, to test a restart strategy
- **`shiftsched_lns.py`**, to test a LNS

For the "LNS" case, the tricky configuration has already been done:

- Try and improve the inner search configuration
  - Change the search strategy
  - Change the time/branch limit
  - There is al a **`SolutionsLimit`**, if you want...
  - ...To stop an iteration when a new solution is found
  - Add restarts (why not?)

Restarts and LNS for Shift-Scheduling

Let's apply these techniques to our shift-scheduling problem

In the start kit, you will find two templates:

- **`shiftsched_restart.py`**, to test a restart strategy
- **`shiftsched_lns.py`**, to test a LNS

For the "LNS" case, the tricky configuration has already been done:

- Try and tweak the fragment selection
  - A random fragment selector has already been implemented
  - Try and change the number of relaxed variables
  - Write a new fragment sel. strategy
  - E.g. relax some employees, relax some days...

Restarts and LNS for Shift-Scheduling

Let's apply these techniques to our shift-scheduling problem

In the start kit, you will find two templates:

- **`shiftsched_restart.py`**, to test a restart strategy
- **`shiftsched_lns.py`**, to test a LNS

There are three benchmark instances:

- **`Instance1.json`**, with optimum 607
- **`Instance2.json`**, with optimum 828
- **`Instance3.json`**, with optimum 1001

See how close you can get

- The main point is assessing the improvement w.r.t. basic DFS
- For this reason, a baseline **`shiftsched_dfs.py`** is provided