This is a example of repeated two-stage SP.

1. Import all the needed data.

```
In[616]:= (* Definition of functions to work on simulation part. *)
          wData =
            Import["wt_data.csv", "CSV", Path → NotebookDirectory[], "HeaderLines" → 1];
          solarRadi = Import["solar.txt", "Table", Path → NotebookDirectory[]];
          markovChainMatrix = {{0.7, 0.2, 0.1}, {0.6, 0.3, 0.1}, {0.5, 0.4, 0.1}};
          stateProb = markovChainMatrix[[1]].MatrixPower[markovChainMatrix, 10];
          randomStateSequence =
            RandomChoice[stateProb → {"N", "A", "M"}, Dimensions[wData][[1]]];

          (* Extract wind and time data *)windSpeed = wData[[All, 20]] / 10 * 0.44704;
          month =
            DateString[{wData[[10, 1]], {"Year", "Month", "Day", " ", "Hour", ":", "Minute"}},
              "Month"] // ToExpression;

          (* Define Price of buying and selling. *)
          BuyPrice[t_] := (
             If[7 ≤ Mod[t, 24] < 11 || 17 < Mod[t, 24] ≤ 21,
              0.099, If[11 ≤ Mod[t, 24] ≤ 17, 0.081,
               0.051]]
            );
          SellPrice[t_] := 0.8 * BuyPrice[t];

          (* Initialize stages, currently two stages. *)
          stageSize = 2;
          availableResources = Table[0, {i, stageSize}];
          demand = Table[0, {i, stageSize}];

          (* Initial values to parameters. *)
          airDensity = 1.27;
          windTurbineCoefficientFactor = 0.5;
          windTurbineRadius = 5;
          solarRadiationFactor = 1.2;
          solarPanelSize = 100;
          initBatt = 100;
          t = 100;
          capBat = 100;

          (* Function of Available Resources at time
           t: electricity power from wind energy and solar energy *)
          AvailableResource[t_] := (
             0.5 * windTurbineCoefficientFactor *
               windTurbineRadius^2 * airDensity * Pi * windSpeed[[t]]^3 +
              solarRadiationFactor * solarPanelSize * solarRadi[[Mod[t, 24] + 1, month]]
            );
```

```
(* Function of Local Demand at time
 t: pick a random integer from 1000 to 10000 *)
Demand[t_] := (
   (*If[t==18,100000,AvailableResource[t]+1000]*)
   2000
   (*RandomInteger[{1000,10000}]*)
  );
(* Function of unit cost of reserving
 electricity in battery : set it constant *)
ReserveCost[t_] := (0.02);
(* Function of unit cost of transiting
 electricity to or from battery : set it constant *)
TransitionCost[t_] := (0.01);

(* Function of optimization of simple two-stage *)
StageMinimize[t_, initBatt_] := (
   Minimize[
    {
     (xGB + xGC) * BuyPrice[t] +
      (initBatt + xGB + xRB - xBC - xBG) * ReserveCost[t] +
      (initBatt + xGB + xRB + xBC + xBG) * TransitionCost[t] -
      (xBG + xRG) * SellPrice[t] +

      If[randomStateSequence[[t]] == "N", markovChainMatrix[[1, 1]],
        If[randomStateSequence[[t]] == "A", markovChainMatrix[[2, 1]],
         markovChainMatrix[[3, 1]]]] * (
           (yGB1 + yGC1) * BuyPrice[t + 1] +
            ((initBatt + xGB + xRB - xBC - xBG) + yGB1 + yRB1 - yBC1 - yBG1) *
          ReserveCost[t + 1] +
            ((initBatt + xGB + xRB - xBC - xBG) + yGB1 + yRB1 + yBC1 + yBG1) *
          TransitionCost[t + 1] -
            (yBG1 + yRG1) * SellPrice[t + 1]) +

      If[randomStateSequence[[t]] == "N", markovChainMatrix[[2, 1]],
        If[randomStateSequence[[t]] == "A", markovChainMatrix[[2, 2]],
         markovChainMatrix[3, 2]]] * (
           (yGB2 + yGC2) * BuyPrice[t + 1] +
            ((initBatt + xGB + xRB - xBC - xBG) + yGB2 + yRB2 - yBC2 - yBG2) *
          ReserveCost[t + 1] +
            ((initBatt + xGB + xRB - xBC - xBG) + yGB2 + yRB2 + yBC2 + yBG2) *
          TransitionCost[t + 1] -
                (yBG2 + yRG2) * SellPrice[t + 1]) +

      If[randomStateSequence[[t]] == "N", markovChainMatrix[[3, 1]],
        If[randomStateSequence[[t]] == "A", markovChainMatrix[[3, 2]],
         markovChainMatrix[[3, 3]]]] * (
           (yGB3 + yGC3) * BuyPrice[t + 1] +
            ((initBatt + xGB + xRB - xBC - xBG) + yGB3 + yRB3 - yBC3 - yBG3) *
          ReserveCost[t + 1] +
```

```
              ((initBatt + xGB + xRB - xBC - xBG) + yGB3 + yRB3 + yBC3 + yBG3) *
           TransitionCost[t + 1] -
              (yBG3 + yRG3) * SellPrice[t + 1]),

      xBC + xGC + xRC == Demand[t] &&
       0 ≤ initBatt + xRB + xGB - xBG - xBC ≤ capBat &&
       xRG + xRB + xRC == AvailableResource[t] &&
       xGB ≥ 0 && xGC ≥ 0 && xRB ≥ 0 && xRC ≥ 0 && xRG ≥ 0 && xBC ≥ 0 && xBG ≥ 0 &&


       yBC1 + yGC1 + yRC1 == Demand[t + 1] &&
       - (initBatt + xGB + xRB - xBC - xBG) ≤
        yRB1 + yGB1 - yBG1 - yBC1 ≤ capBat - (initBatt + xGB + xRB - xBC - xBG) &&
       yRG1 + yRB1 + yRC1 == AvailableResource[t + 1] &&
       yGB1 ≥ 0 && yGC1 ≥ 0 && yRB1 ≥ 0 && yRC1 ≥ 0 && yRG1 ≥ 0 && yBC1 ≥ 0 && yBG1 ≥ 0 &&

       yBC2 + yGC2 + yRC2 == Demand[t + 1] &&
       - (initBatt + xGB + xRB - xBC - xBG) ≤
        yRB2 + yGB2 - yBG2 - yBC2 ≤ capBat - (initBatt + xGB + xRB - xBC - xBG) &&
       yRG2 + yRB2 + yRC2 == AvailableResource[t + 1] / 2 &&
       yGB2 ≥ 0 && yGC2 ≥ 0 && yRB2 ≥ 0 && yRC2 ≥ 0 && yRG2 ≥ 0 && yBC2 ≥ 0 && yBG2 ≥ 0 &&

       yBC3 + yGC3 + yRC3 == Demand[t + 1] &&
       - (initBatt + xGB + xRB - xBC - xBG) ≤
        yRB3 + yGB3 - yBG3 - yBC3 ≤ capBat - (initBatt + xGB + xRB - xBC - xBG) &&
       yRG3 + yRB3 + yRC3 == AvailableResource[t + 1] / 4 &&
       yGB3 ≥ 0 && yGC3 ≥ 0 && yRB3 ≥ 0 && yRC3 ≥ 0 && yRG3 ≥ 0 && yBC3 ≥ 0 && yBG3 ≥ 0
     },
     {xGB, xGC, xRB, xRC, xRG, xBC, xBG, yGB1, yGC1, yRB1, yRC1, yRG1, yBC1, yBG1, yGB2,
      yGC2, yRB2, yRC2, yRG2, yBC2, yBG2, yGB3, yGC3, yRB3, yRC3, yRG3, yBC3, yBG3}]
  );
```

2. Run it once and get the result of the optimized first stage.

```
In[641]:= firstStage = StageMinimize[t, initBatt]

Out[641]= {17.9696, {xGB → 0., xGC → 0., xRB → 0., xRC → 2000., xRG → 364.14,
            xBC → 0., xBG → 100., yGB1 → 0., yGC1 → 0., yRB1 → 0., yRC1 → 2000.,
            yRG1 → 295.286, yBC1 → 0., yBG1 → 0., yGB2 → 0., yGC2 → 852.357, yRB2 → 0.,
            yRC2 → 1147.64, yRG2 → 0., yBC2 → 0., yBG2 → 0., yGB3 → 0., yGC3 → 1426.18,
            yRB3 → 0., yRC3 → 573.822, yRG3 → 0., yBC3 → 1.13687 × 10^-13, yBG3 → 0.}}
```

3. Reset the battery amount

```
In[642]:= initBatt = N[(xRB /. firstStage[[2]]) + (xGB /. firstStage[[2]]), 2]

Out[642]= 0.
```

4. Run it again, scenario at time t+1 is already simulated in randomStateSequence.

```
In[643]:= secondStage = StageMinimize[t + 1, initBatt]

Out[643]= {41.9686, {xGB → 0., xGC → 0., xRB → 0., xRC → 2000., xRG → 295.286, xBC → 0., xBG → 0.,
  yGB1 → 0., yGC1 → 0., yRB1 → 0., yRC1 → 2000., yRG1 → 295.286, yBC1 → 0., yBG1 → 0.,
  yGB2 → 0., yGC2 → 852.357, yRB2 → 0., yRC2 → 1147.64, yRG2 → 0., yBC2 → 0., yBG2 → 0.,
  yGB3 → 0., yGC3 → 1426.18, yRB3 → 0., yRC3 → 573.822, yRG3 → 0., yBC3 → 0., yBG3 → 0.}}

$MachinePrecision

15.9546

$MachineEpsilon

2.22045 × 10^-16
```

A repeated two-stage example.

```
In[644]:= Table[StageMinimize[t, initBatt][[1]], {t, 1, 10, 1}]
          (* ignore the initial battery amount change at each step *)
```

```
NMinimize::incst :
  NMinimize was unable to generate any initial points satisfying the inequality constraints {-2000. + 1. xBC + 1. xGC ≤ 0, 356.771
            - 3.33067×10^-16 xBC - xBG + xGB + 1. xGC - 1. xRG ≤ 0, -456.771 - 1. xBC - 1. xGC + 1. xRG ≤ 0, ≪10≫, 1870.4
            + 1. yGC3 - 1. yRB3 - 1. yRG3 ≤ 0, -3870.4 + 1. yRB3 + 1. yRG3 ≤ 0, -2327.17 + 3.33067×10^-16 xBC + xBG - xGB - 1.
              xGC + 1. xRG + yBG3 - yGB3 - 1. yGC3 + 1. yRG3 ≤ 0}. The initial region specified may not
     contain any feasible points. Changing the initial region or specifying explicit initial points may provide a better solution. ≫

NMinimize::nnum : The function value 90.8946 + 70.8838 {{0.7, 0.2, 0.1}, {0.6, 0.3, 0.1}, {0.5, 0.4, 0.1}}[3, 2] is not a number at
     {xBC, xBG, xGB, xGC, xRB, xRC, xRG, yBC1, yBC2, yBC3, yBG1, yBG2, yBG3, yGB1, yGB2, yGB3, yGC1, yGC2, yGC3, yRB1, yRB2,
       yRB3, yRC1, yRC2, yRC3, yRG1, yRG2, yRG3} =
     {0.0282287, 1.97318, 0.0367799, 0.0107591, 456.805, 1999.96, 0.00467042, -13479.6, -5738.82, -1868.44, 0.03725,
       0.0827363, 0.0155209, ≪21≫, ≪20≫, ≪21≫, 0.0295334, 0.0288056, 0.00178635, 0.0463451, 0.0600338, 0.00815288,
       15479.6, 7738.79, 3868.44, 1.96662, 1.95585, 1.95572}. ≫
```

```
Out[644]= {32.8356, 42.5504, 48.1138, 51.4377, 33.2892,
           47.4933, {0.099 (xGB + xGC) + 0.02 (0. - xBC - xBG + xGB + xRB) +
             0.01 (0. + xBC + xBG + xGB + xRB) - 0.0792 (xBG + xRG) +
             0.5 (0.099 (yGB1 + yGC1) + 0.02 (0. - xBC - xBG + xGB + xRB - yBC1 - yBG1 + yGB1 + yRB1) +
                0.01 (0. - xBC - xBG + xGB + xRB + yBC1 + yBG1 + yGB1 + yRB1) - 0.0792 (yBG1 + yRG1)) +
             0.1 (0.099 (yGB3 + yGC3) + 0.02 (0. - xBC - xBG + xGB + xRB - yBC3 - yBG3 + yGB3 + yRB3) +
                0.01 (0. - xBC - xBG + xGB + xRB + yBC3 + yBG3 + yGB3 + yRB3) - 0.0792 (yBG3 + yRG3)) +
             (0.099 (yGB2 + yGC2) + 0.02 (0. - xBC - xBG + xGB + xRB - yBC2 - yBG2 + yGB2 + yRB2) +
                0.01 (0. - xBC - xBG + xGB + xRB + yBC2 + yBG2 + yGB2 + yRB2) - 0.0792 (yBG2 + yRG2))
              {{0.7, 0.2, 0.1}, {0.6, 0.3, 0.1}, {0.5, 0.4, 0.1}}[3, 2],
            xBC + xGC + xRC == 2000 && 0 ≤ 0. - xBC - xBG + xGB + xRB ≤ 100 &&
             xRB + xRC + xRG == 2456.77 && xGB ≥ 0 && xGC ≥ 0 && xRB ≥ 0 &&
             xRC ≥ 0 && xRG ≥ 0 && xBC ≥ 0 && xBG ≥ 0 && yBC1 + yGC1 + yRC1 == 2000 &&
             0. + xBC + xBG - xGB - xRB ≤ - yBC1 - yBG1 + yGB1 + yRB1 ≤ 100. + xBC + xBG - xGB - xRB &&
             yRB1 + yRC1 + yRG1 == 15 481.6 && yGB1 ≥ 0 && yGC1 ≥ 0 && yRB1 ≥ 0 &&
             yRC1 ≥ 0 && yRG1 ≥ 0 && yBC1 ≥ 0 && yBG1 ≥ 0 && yBC2 + yGC2 + yRC2 == 2000 &&
             0. + xBC + xBG - xGB - xRB ≤ - yBC2 - yBG2 + yGB2 + yRB2 ≤ 100. + xBC + xBG - xGB - xRB &&
             yRB2 + yRC2 + yRG2 == 7740.81 && yGB2 ≥ 0 && yGC2 ≥ 0 && yRB2 ≥ 0 &&
             yRC2 ≥ 0 && yRG2 ≥ 0 && yBC2 ≥ 0 && yBG2 ≥ 0 && yBC3 + yGC3 + yRC3 == 2000 &&
             0. + xBC + xBG - xGB - xRB ≤ - yBC3 - yBG3 + yGB3 + yRB3 ≤ 100. + xBC + xBG - xGB - xRB &&
             yRB3 + yRC3 + yRG3 == 3870.4 && yGB3 ≥ 0 && yGC3 ≥ 0 && yRB3 ≥ 0 && yRC3 ≥ 0 &&
             yRG3 ≥ 0 && yBC3 ≥ 0 && yBG3 ≥ 0}, -2522.46, -4440.23, -4362.97}
```

The example shows above is the example of generating repeated two-stage in 10 steps. But in the one of the step, it cannot get the optimized solution.

The result is short for like this:
{32.8356,42.5504,48.1138,51.4377,33.2892,47.4933,***,-2522.46,-4440.23,-4362.97}

Also notice in two-stage the result for the first step contains one optimized variable like {yBC3->1.13687*10^-13**}, which is obvious quite small, and such kind of variable value only appears in**

Battery to Consumer (BC).