INPUT: T\_out\_pred, RH, u, alpha1, beta, target\_objective

OUTPUT: T\_opt, total\_energy\_demand, total\_smoothness

Initialize Parameters:

- Set constants: A, eta, epsilon, L, U

Define Functions:

FUNCTION calculate\_etis(T, RH, u):

RETURN computed ETIS

FUNCTION calculate\_energy\_demand(T, RH, u, i):

energy\_demand = (A / eta) \* ((T[i] - epsilon \* T[i-1]) / (1 - epsilon) - T\_out\_pred[i-1])

IF energy\_demand < 0 THEN:

energy\_demand = 0

T[i] = adjust towards T\_out\_pred[i-1]

RETURN energy\_demand

FUNCTION objective(T, RH, u, energy\_demand\_values, smooth\_values):

etis\_values = calculate\_etis(T, RH, u)

FOR i FROM 1 TO length(T):

energy\_demand\_values[i] = calculate\_energy\_demand(T, RH, u, i)

FOR i FROM 1 TO length(etis\_values):

smooth\_values[i] = (etis\_values[i] - etis\_values[i - 1])^2

E\_min = min(energy\_demand\_values)

E\_max = max(energy\_demand\_values)

S\_min = min(smooth\_values)

S\_max = max(smooth\_values)

IF E\_min != E\_max THEN:

percent\_E = 100 \* (1 - (energy\_demand\_values - E\_min) / (E\_max - E\_min))

ELSE:

percent\_E = array filled with 100 of length T

IF S\_min != S\_max THEN:

percent\_S = 100 \* (1 - (smooth\_values - S\_min) / (S\_max - S\_min))

ELSE:

percent\_S = array filled with 100 of length T

objective\_values = alpha1 \* percent\_S + beta \* percent\_E

RETURN objective\_values, percent\_S, percent\_E

Optimize Component Per Step:

FUNCTION optimize\_component\_per\_step\_with\_etis\_constraint(func, T\_initial, RH, u, T\_out\_pred, learning\_rate, num\_steps):

T\_opt = copy of T\_initial

FOR step IN range(num\_steps):

FOR i IN range(1, length(T\_opt)):

loss\_minus = func(T\_opt with T[i] - delta)

loss\_plus = func(T\_opt with T[i] + delta)

gradient = (loss\_plus - loss\_minus) / (2 \* delta)

T\_opt[i] -= learning\_rate \* gradient

Adjust T\_opt[i] to satisfy ETIS constraints

RETURN T\_opt

Optimize Temperatures with Objective:

FUNCTION optimize\_temperatures\_with\_objective(RH, u, T\_out\_pred):

T\_initial = array filled with initial temperature

energy\_demand\_values = array of zeros with length T\_initial

smooth\_values = array of zeros with length T\_initial

T\_opt\_energy = optimize\_component\_per\_step\_with\_etis\_constraint(calculate\_energy\_demand, T\_initial, RH, u, T\_out\_pred, learning\_rate, num\_steps)

T\_smoothness\_values = optimize\_component\_per\_step\_with\_etis\_constraint(calculate\_energy\_demand, T\_initial, RH, u, T\_out\_pred, learning\_rate, num\_steps)

objective\_values, percent\_S, percent\_E = objective(T\_opt\_energy, RH, u, energy\_demand\_values, smooth\_values)

RETURN T\_opt\_energy, T\_smoothness\_values, objective\_values

Compute Results:

total\_energy\_demand = sum of energy demands for T\_opt\_energy

total\_smoothness = compute smoothness of ETIS for T\_smoothness\_values

RETURN T\_opt\_energy, total\_energy\_demand, total\_smoothness