```python
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import solve_bvp

# set up the right hand side of the ODE, with the parameter
# tf (the final time rescaled)
def chemo_finite_time(t,y,p):
    alpha = 1
    tf=p[0]
    return np.vstack((tf*alpha*y[0] + tf*0.5*y[1], -tf*alpha*y[1]))

# set up the endpoint conditions.  The final one is H(tf) = 2a(tf-1)
def bc_new(ya,yb,p):
    x0 = 2
    a = 1
    alpha = 1
    m = 2
    tf = p[0]
    return np.array([ya[0]-x0, yb[1]+m, alpha*yb[0]*yb[1]+.25*yb[1]**2-2*a*(tf-1)])

# construct the solution using solve_bvp.  We use an initial guess
# of tf=1 as that is the value we are forcing the solution to be close to.
t=np.linspace(0,1,10)
y = np.zeros((2, t.size))
res = solve_bvp(chemo_finite_time, bc_new, t, y, p=[1])

# plot the results and print out the optimal final time t_f
t_plot = np.linspace(0, 1, 100)
x_plot = res.sol(t_plot)[0]
p_plot = res.sol(t_plot)[1]
plt.plot(t_plot, x_plot)
plt.plot(t_plot, -.5*p_plot)
plt.legend(['cancer cells','chemo concentration'])
print('t_f = '+str(res.p[0]))
```

Algorithm 21.2: Algorithm for computing the solution of the Chemotherapy problem in Example 21.3.2 for specific values of the relevant parameters.