```
import simpy, numpy as np
from pyDOE2 import lhs
from scipy.optimize import basinhopping
import matplotlib.pyplot as plt

class AssemblyLine:
    def __init__(self, params):
        self.params = params
        self.env = simpy.Environment()
        self.cost = 0

    def simulate(self):
        res = simpy.Resource(self.env, capacity=int(self.params['capacity']))
        self.cost += self.params['capacity'] * 100
        for _ in range(int(self.params['num_products'])):
            with res.request() as req:
                yield req
                t = 1 / (self.params['prod_rate'] * self.params['efficiency'])
                yield self.env.timeout(t)
                self.cost += t * self.params['worker_rate'] * 10 * self.params['efficiency']
                self.cost += 200 / (self.params['prod_rate'] * self.params['efficiency'] * self.params['worker_rate'])

    def run(self):
        self.env.process(self.simulate())
        self.env.run()
        return self.cost

def objective(p): return AssemblyLine(p).run()

def generate_samples(n):
    bounds = [(50,200), (5,15), (50,200), (0.8,1.2)]
    s = lhs(4, samples=n)
    return [{'prod_rate': b[0]+x*(b[1]-b[0]), 'worker_rate': b2[0]+x2*(b2[1]-b2[0]),
             'capacity': b3[0]+x3*(b3[1]-b3[0]), 'efficiency': b4[0]+x4*(b4[1]-b4[0]), 'num_products':200}
            for (x,x2,x3,x4),b,b2,b3,b4 in zip(s,bounds,bounds,bounds,bounds)]

def fit_model(X, y):
    A = np.column_stack([np.ones(len(X))]+[X[:,i] for i in range(4)]+[X[:,i]**2 for i in range(4)]+
                        [X[:,i]*X[:,j] for i in range(4) for j in range(i+1,4)])
    return np.linalg.lstsq(A, y, rcond=None)[0]

def predict(c, x): return np.dot(c, [1]+list(x)+[i**2 for i in x]+[x[i]*x[j] for i in range(4) for j in range(i+1,4)])

def optimize():
    data = generate_samples(100)
    X = np.array([[d['prod_rate'], d['worker_rate'], d['capacity'], d['efficiency']] for d in data])
    y = np.array([objective(d) for d in data])
    coeffs = fit_model(X, y)
    obj = lambda x: predict(coeffs, x)
    result = basinhopping(obj, [100, 10, 100, 1], minimizer_kwargs={'bounds': [(50,200),(5,15),(50,200),(0.8,1.2)]})
    best = result.x
    print("Optimized Settings:")
    for k,v in zip(['prod_rate','worker_rate','capacity','efficiency'], best): print(f"{k}: {v:.2f}")
    print("Estimated Cost:", result.fun)
    plot(X, y, coeffs, best)

def plot(X, y, coeffs, best):
    pr, wr = np.linspace(50,200,100), np.linspace(5,15,100)
    PR, WR = np.meshgrid(pr, wr)
    Z = np.array([predict(coeffs, [p,w,best[2],best[3]]) for p,w in zip(PR.ravel(), WR.ravel())]).reshape(PR.shape)
    plt.contourf(PR, WR, Z, 20, cmap='viridis')
    plt.colorbar(label='Cost')
    plt.scatter(X[:,0], X[:,1], c=y, cmap='jet', label='Samples')
    plt.scatter(best[0], best[1], c='red', marker='x', label='Optimized')
    plt.xlabel('Production Rate'); plt.ylabel('Worker Rate')
    plt.title('Response Surface'); plt.legend(); plt.show()

optimize()
```