```python
def perturb(x, alpha=0.1):
    # generate random point in the unit hypersphere
    ndim = x.shape[0]
    random_point = np.random.normal(size=ndim)
    random_point /= np.linalg.norm(random_point)

    # scale and translate the point to fit the specified center and radius
    perturbed_point = x + alpha * x * random_point

    return perturbed_point

def b_mod(costf=None, parameters=None, alpha=0.1, iterations=2, N=100, M=10, max_time=-1, debug=False):
    if costf is None:
        raise Exception("costf is a required kwarg")
    if parameters is None:
        raise Exception("parameters is a required kwarg")
    it_best_costs = []
    start_time = time.time()
    best_cost = None
    best_params = None
    times = []
    if max_time > 0:
        N = -1
    current_time = 0
    params = []
    costs = []
    it = 0
    while (it < N or N < 0) and (current_time < max_time or max_time < 0):
        it += 1
        ps = gen_params(parameters)
        cost = costf(ps)
        params.append(ps)
        costs.append(cost)
        if best_cost is None or cost < best_cost:
            best_params = ps
            best_cost = cost
        it_best_costs.append(best_cost)
        current_time = time.time() - start_time
        times.append(current_time)
        if debug:
            print("parameters:", ps, end="\t")
            print("cost:", cost, end="\t")
            print("best cost:", best_cost)
    bests = best_m(params, costs, M=M)

    for i in range(iterations):
        params = []
        costs = []
        it = 0
        while it < N and (current_time < max_time or max_time < 0):
            it += 1
            choice = random.choice(bests)
            new_params = perturb(choice, alpha=alpha)
            new_cost = costf(choice)
            params.append(new_params)
            costs.append(new_cost)
            if new_cost < best_cost:
                best_cost = new_cost
                best_params = new_params

    return {
        "results": {
            "best_params": best_params,
            "best_cost": best_cost,
        },
        "stats": {
            "it_best_costs": it_best_costs,
            "time": times,
        }
    }
```