MTRN4030

Optimisation Methods for Engineering Systems

Laboratory Exercise 4 – weeks 9, 10, 11

Metaheuristics and Matlab Programming

Summary

This laboratory instruction sheet contains two parts. Part 1 is a tutorial of the metaheuristic optimization methods covered in the course. Part 2 is about the exercise on the use of Matlab to develop programs to solve, in the computer, optimization problems using metaheuristic approaches. No written report is required for this laboratory. On the other hand, the developed program has to be shown to tutors and marked-off at the end of the scheduled laboratory class (see Assessment section below).

Part 1 - Tutorial

Heuristics is a solution strategy by trial-and-error to produce **acceptable** solutions to a complex problem in a reasonably practical time. The complexity of the problem of interest makes it impossible to search for every possible solution or combination, the aim is to find good, feasible solutions in an acceptable **timescale**. There is no guarantee that the best solutions can be found, and we even do not know whether an algorithm will work and why if it does work. The idea is that an efficient and practical algorithm that will work most of the time and be able to produce good quality solutions. Among the found quality solutions, it is expected that some of them are **nearly optimal**, though there is **no guarantee for optimality**.

Random Walk

A random walk is a random process that consists of taking a series of consecutive random steps. Mathematically speaking, let S_N denotes the sum of each consecutive random step X_i, then S_N forms a random walk

$$S_N = \sum_{i=1}^{N} X_i = X_1 + X_2 + \cdots + X_N$$

where X_i is a random step drawn from a random distribution. This relationship can also be written as a recursive formula

$$S_N = \sum_{i=1}^{N-1} X_i + X_N = S_{N-1} + X_N$$

which means that the next state S_N will only depend on the currently existing state S_{N-1} and how the motion or transition X_N from the existing state to the next state. Here the step size or length in a random walk can be fixed or varying. This is typically the main property of a Markov chain.

Simulated Annealing

Simulated annealing is a random search technique for global optimization problems, and it mimics the annealing process in material processing when the metal cools and freezes into a crystalline state with the minimum energy and larger crystal size so as to reduce the defects in metallic structures. The annealing process involves the careful control of temperature and cooling rate, often called annealing schedule.

The basic idea of the simulated annealing algorithm is to use random search in terms of a Markov chain, which not only accepts changes that improve the objective function, but also keeps some changes that are not ideal.

In a minimization problem, for example, any better moves or changes that decrease the value of the objective function f will be accepted; however, some changes that increase f will also be accepted with a probability p. This probability p, also called the transition probability, is determined by

$$p = exp\left(\frac{-\Delta E}{k_B T}\right)$$

where k_B is the Boltzmann's constant, and for simplicity, we can use k to denote k_B because $k = 1$ is often used. T is the temperature for controlling the annealing process. ΔE is the change in energy levels. Whether or not we accept a change, we usually use a random number τ as a threshold. Thus, if $p > \tau$, the move is accepted.

Genetic Algorithm

The Genetic Algorithm (GA), is a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection. The essence of GA involves the encoding of an optimization function as arrays of bits or character strings to represent the chromosomes, the manipulation operations of strings by genetic operators, and the selection according to their fitness with the aim to find a solution to the problem concerned. This is often done by the following procedure:

1. Encoding the objectives or optimization functions;
2. Defining a fitness function or selection criterion;
3. Initializing a population of individuals;
4. Evaluating the fitness of all the individuals in the population;
5. Creating a new population by performing crossover, and mutation, fitness proportionate reproduction etc;
6. Evolving the population until certain stopping criteria are met;
7. Decoding the results to obtain the solution to the problem.

The crossover of two parent-strings is the main operator with a higher probability p_c and is carried out by swapping one segment of one chromosome with the corresponding segment on another chromosome at a random position. The mutation operation is achieved by flopping the randomly selected bits, and the mutation probability p_m is usually small. The selection of an individual in a population is carried out by the evaluation of its fitness, and it can remain in the new generation if a certain threshold of the fitness is reached, we can also use that the reproduction of a population is fitness-proportionate. That is to say, those individuals with higher fitness are more likely to reproduce. For example, we can use the individual fitness assignment relative to the whole population

$$F(x_i) = \frac{f(\zeta_i)}{\sum_{i=1}^{N} f(\zeta_i)}$$

where N is the population size. Here ζ_i is the phenotypic value of individual i, that is, the solution represented by an individual chromosome. The appropriate form of the fitness function will make sure that the solutions with higher fitness should be selected efficiently.

Particle Swarm Optimization

Particle swarm optimization is based on swarm behaviour such as fish and bird schooling in nature. It uses the real-number randomness and global communication among the swarm particles. In this sense, it is also easier to implement as there is no encoding or decoding of the parameters into binary strings such as those in Genetic Algorithms which can also use real-number strings. This algorithm searches the space of an objective function by adjusting the trajectories of individual agents, called particles, as these trajectories form piecewise paths in a quasi-stochastic manner.

The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best $\boldsymbol{g}^*$and its own best location $\boldsymbol{x}_i^*$ in history, while at the same time it has a tendency to move randomly. When a particle finds a location that is better than any previously found locations, then it updates it as the new current best for particle i.

Let $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ be the position vector and velocity for particle i, respectively. The new velocity vector is determined by

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha\epsilon_1 \otimes \left(\boldsymbol{g}^* - \boldsymbol{x}_i^t\right) + \beta\epsilon_2 \otimes (\boldsymbol{x}_i^* - \boldsymbol{x}_i^t)$$

where ϵ_1 and ϵ_2 are two random vectors, and each entry taking the values between 0 and 1, and $(\boldsymbol{u} \otimes \boldsymbol{v})_{ij} = u_{ij}v_{ij}$.

The initial locations of all particles should distribute relatively uniformly so that they can sample over most regions, which is especially important for multimodal problems. The initial velocity of a particle can be taken as zero, that is, $\boldsymbol{v}_i^{t=1} = 0$. The new position can then be updated by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}$$

Ant Colony Optimization

Ants are social insects in habitation and they live together in organized colonies whose population size can range from about 2 to 25 million. When foraging, a swarm of ants or mobile agents interact or communicate in their local environment. Each ant can lay scent chemicals or pheromone so as to communicate with others, and each ant is also able to follow the route marked with pheromone laid by other ants. When ants find a food source, they will mark it with pheromone and also mark the trails to and from it. From the initial random foraging route, the pheromone concentration varies and the ants follow the route with higher pheromone concentration, and the pheromone is enhanced by the increasing number of ants. As more and more ants follow the same route, it becomes the favoured path. Thus, some favourite routes emerge, often the shortest or more efficient.

Apart from the population size, there are two important issues here: the probability of choosing a route and the evaporation rate of pheromone. For a network routing problem, the probability of ants at a particular node *i* to choose the route from node *i* to node *j*, among n_d nodes, is given by

$$p_{ij} = \frac{\phi_{ij}^{\alpha} d_{ij}^{\beta}}{\sum_{i,j=1}^{n_f} \phi_{ij}^{\alpha} d_{ij}^{\beta}}$$

where $\alpha, \beta > 0$ are the influence parameters. ϕ_{ij} is the pheromone concentration on the route between i and j , and $d_{ij}$the desirability of the same route. Some *a priori* knowledge about the route such as the distance s_{ij} is often used so that $d_{ij} \propto 1/s_{ij}$, which implies that shorter routes will be selected due to their shorter travelling time, and thus the pheromone concentrations on these routes are higher.

For a constant rate γ of pheromone decay or evaporation, the pheromone concentration usually varies with time exponentially

$$\phi(t) = \phi_0 e^{-\gamma t}$$

where ϕ_0 is the initial concentration of pheromone and t is time. If $\gamma \ll 1$ then we have $\phi(t) \approx (1 - \gamma t)\phi(0)$. For the unitary time increment $\Delta t = 1$ the evaporation can be approximated by $\phi^{t+1} = (1 - \gamma)\phi^t$. Therefore, we have the simplified pheromone update formula:

$$\phi^{t+1} = (1 - \gamma)\phi_{ij}^t + \delta\phi_{ij}^t$$

where $\gamma \in [0,1]$ is the rate of pheromone evaporation. The increment $\delta\phi_{ij}^t$ is the amount of pheromone deposited at time t along route i to j when an ant travels a distance L. Usually $\delta\phi_{ij}^t \propto 1/L$. If there are no ants on a route, then the pheromone deposit is zero.

Multiobjective Optimization

In reality, we often have to optimize multiple objectives simultaneously. For example, we may want to improve the performance of a product while trying to minimize the cost at the same time. In this case, we are dealing with multiobjective optimization problems.

Any multiobjective optimization problem can generally be written as

$$maximize_{x \in R}\, \boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_p(\boldsymbol{x})]$$

$$subject\ to\ g_j(\boldsymbol{x}) \leq 0, j = 1,2, \cdots, M$$

$$h_k(\boldsymbol{x}) = 0, k = 1,2, \cdots, N$$

where $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$ is the vector of decision variables.

Many solution algorithms intend to combine all the multi-objective functions into one scalar objective using the weighted sum

$$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x}) + \cdots + \alpha_p f_p(\boldsymbol{x})$$

The important issue arises in assigning the weighting coefficients $(\alpha_1, \alpha_2, \cdots, \alpha_p)$ because the solution strongly depends on the chosen weighting coefficients.

For Pareto optimality, a vector $\boldsymbol{u} = [u_1, u_2, \cdots, u_n]^T$ is said to dominate another vector $\boldsymbol{v} = [v_1, v_2, \cdots, v_n]^T$ if and only if $u_i \leq v_i$ for all i such that $\forall i \in \{1,2, \cdots, n\}$ and there is an i such that $\exists i \in \{1,2, \cdots, n\}: u_i < v_i$. A point or a solution $\boldsymbol{x}^* \in R^n$ is called a Pareto optimal solution or non-inferior solution to the optimization problem if there is no $\boldsymbol{x}$ satisfying $f_1(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*), i = 1,2, \cdots, p$. That is to say, optimal solutions are solutions which are not dominated by any other solutions.

Unlike the single-objective optimization with often a single optimal solution, multiobjective optimization will lead to a set of solutions, called the Pareto optimal set $\mathcal{P}^*$, and the

decision vectors $\boldsymbol{x}^*$ for this solution set are thus called non-dominated. The plot of this Pareto set in the objective or response space is called the Pareto front.

Utility method, on the other hand, considers uncertainty in the criteria values for each alternative, which is a more realistic method because there is always some degree of uncertainty about the outcome of a particular alternative.

The utility function defines combinations of objective values $f_1, f_2, \cdots, f_p$, which a decision-maker finds equally acceptable or indifference. So the contours of the constant utility are referred to as the indifference curves. The optimization now becomes the maximization of the utility. Possible utility functions could be

$$U(f_1, f_2) = k f_1^{\alpha} f_2^{\beta}$$

$$U(f_1, f_2) = \alpha f_1 + \beta f_2 + [1 - (\alpha + \beta)] f_1 f_2$$

Part 2 - Exercise

In this exercise, you are required to develop Matlab programs to solve the following engineering design problems. You have to apply different metaheuristic optimization approaches to obtain the solutions. That is, solve the problem in **Task 1** using the real-coded genetic algorithm (GA). For **Task 2**, you have to use the particle swarm optimization (PSO) algorithm. For **Task 3,** use ants colony optimization (ACO) algorithm to solve the problem. For **Task 4**, you are required to use the binary coded Genetic Algorithm.

Task 1 – Costing Problem

It has been decided to shift grain from a warehouse to a factory in an open rectangular box of length x_1 meters, width x_2 meters, and height x_3 meters. The bottom, sides, and the ends of the box cost, respectively, \$20, \$10, and \$20/m^2. It costs \$1 for each round trip of the box. Assume that the box will have no salvage value, find the minimum cost of transporting 100m^3 of grain.

The total cost of transportation is given by

$$total\ cost = cost\ of\ box + cost\ of\ transportation$$

That is

$$f(\boldsymbol{x}) = 20x_1x_2 + 10x_2x_3 + 20x_1x_3 + \frac{100}{x_1x_2x_3}$$

where x_1, x_2, and x_3 indicate the dimensions of the box, as shown in Figure 1. The lengths are limited within $x_1, x_2, x_3 \in [0.2, 5.0]$.

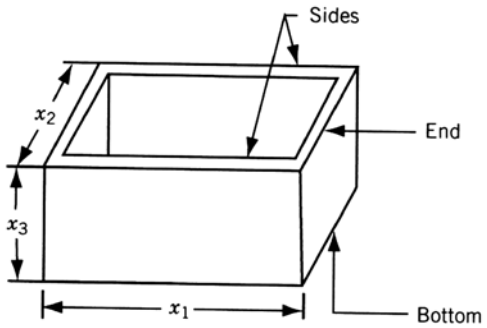

Figure 1. Open rectangular box.

In your program, you have to provide a plot of the minimum cost against iterations. Discuss the variation in solution value f with respect to the number of iterations and the number of agents.

Task 2 - Optimal Control of a Non-Linear Stirred Tank Reactor

A first-order irreversible chemical reaction carried out in a continuous stirred tank reactor is a multimodal optimal control problem. This chemical process is modeled by two non-linear differential equations:

$$\dot{x_1} = -(2+u)(x_1+0.25) + (x_2+0.5)\exp(25x_1/(x_1+2))$$
$$\dot{x_2} = 0.5 - x_2 - (x_2+0.5)\exp(25x_1/(x_1+2))$$

where $u(t)$=flow rate of the cooling fluid, x_1=dimensionless steady state temperature and x_2=Deviation from dimensionless steady state concentration. The optimization objective is to determine suitable value of u so that the objective function

$$J = \int_0^{t_f=0.72} (x_1^2 + x_2^2 + 0.1u^2)dt$$

is minimizes where the initial conditions are $\boldsymbol{x}(0) = [0.09\ 0.09]^T$. The problem is constrained where the range in $u(t)$ lies in $[0.0\ 0.5]$. The integration involved in evaluation is performed using function ode45 available in MATLAB with relative tolerance set to 0.1.

Hint: To obtain the objective function, your Matlab code may include the following.

```
Tol=0.1; % tolerance
Tspan=0.72; % integration limit
X0=[0.09 0.09]; % initial system state
Options=odeset('RelTol',Tol); % option sets for integration
[T,X]=ode45(@(t,x) intgrl(t,x,u),[0 Tspan],X0,Options);
Fit=sum(sum(X.^2,2)+0.1*u.^2);
```

And 'intgrl' is another function that calculates variables x_1 and x_2.

```
function dx = intgrl(t,x,u)
dx=zeros(2,1);
dx(1)=-(2+u)*(x(1)+0.25)+(x(2)+0.5)*exp(25*x(1)/(x(1)+2));
dx(2)=0.5-x(2)-(x(2)+0.5)*exp(25*x(1)/(x(1)+2));
```

In your program, you have to provide a plot of the minimum objective function against iterations. Discuss the variation in the minimum objective function with respect to the number of iterations and the number of agents.

Task 3 – ***Path Planning Problem***

Consider a region of $100 \times 100\ km^2$ and there are T towns (e.g., $T = 50$) with random (but known) locations. A traveller needs to travel from the town located at the most south-west (starting) to the town at the most north-east (target). Find the shortest travel path that the traveller has to visit the largest number of (but not all) towns. That is, the objective function is directly proportional to the number of towns visited and inversely proportional to the total distance travelled.

In your program, you have to provide a plot of the minimum objective function against iterations. Discuss the variation in the minimum objective function with respect to the ACO parameter settings (e.g., α, β, and γ for pheromone update with $\phi^{t+1} = \gamma\phi^t$, $\gamma \in [0,1]$).

The following program procedures may be adopted.

1. Define region, number of towns, town locations, plot town locations with label
2. Compute inter-distances between towns and the inverse-distances (in matrix form)
3. Find and plot starting and target towns
4. Define ACO parameters α, β, γ and initial pheromone (in matrix form)
5. For each generation
    a. Set Travel=0, Visit=0 (in a matrix)
    b. For each ant
        i. Randomly generate a location within the region
        ii. Find the nearest two towns and put into a list of visited towns
        iii. Add the distance to Travel
        iv. While Start or Target town not visited
            1. Calculate travel probability $p = \phi^\alpha d^\beta$
            2. Assign next town from the maximum probability and add to the visited list
            3. Update the Visit matrix corresponding to the town visited
        v. End while
    c. End each ant
    d. Update pheromone matrix by $\phi^t = \phi^t +$ reward (e.g., $\Delta\phi = T_N/D_T$, number of tons visited divided by total distance travelled)
    e. Normalize and evaporate pheromone
    f. Determine the path travelled and calculate the distance travelled
    g. If a better path is found, show the path
6. End each generation
7. Show final path

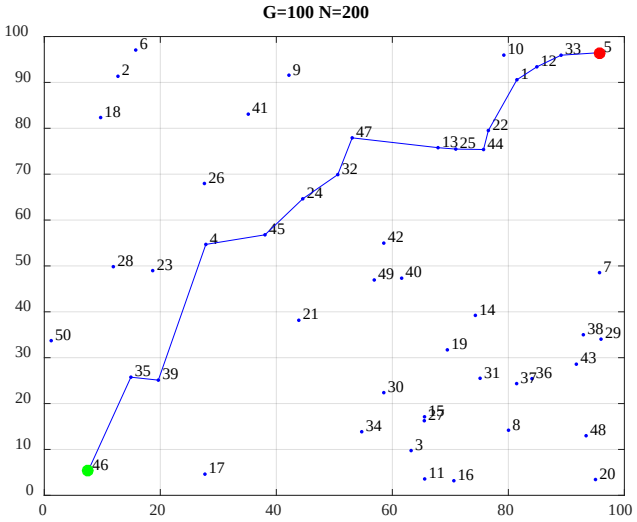

Figure 2. Example result of path planning.

Task 4 – Routing Problem

Consider a traffic routing design over an area of $100 \times 100\ km^2$ (within a coordinate system of $x = 0, \cdots, 100, y = 0, \cdots, 100$. Let there are T towns within this area. The towns will be connected by roads to several traffic hub towns H. The hubs are also connected by roads.

In the planning stage, only the locations of the towns in the area are known. It is to be decided which of the towns will be the traffic hub. The objective is to minimize traffic cost as the sum of: (1) road traffic cost ($2) that is directly proportional to the distance travelled from a town to a hub; (2) the cost incurred in travelling between hubs ($1).

Assume all traffic follow straight lines. In the task, you can specify a reasonable set of parameters, e.g, $T = 10,20,30$, etc. The number of hubs as $H = 2,3,4$, etc.

Hint: For this problem, the binary Genetic Algorithm (GA) would be most applicable. The number of bits in a string would correspond to the number of towns. The hub towns can be represented as a '1' in the bit string while other '0' bits represent the other towns. Assume the travel cost is one unit for each unit distance travelled; the objective function is the total cost including the road and train travel.

The following program procedures may be adopted.

1. Define Town locations, number of Hubs.
2. Define GA parameters: (number of generations, chromosomes, cross-over, and mutation probabilities)
3. Generate initial chromosomes
4. For each generation
    a. For each chromosome
        i. Assign an individual town to a hub
        ii. Determine the correspondence between towns and hubs
        iii. Find cost between hubs
        iv. Find cost from towns to hubs
        v. If cost is minimum
            1. Store town, hub parameters that form the best design
        vi. End

b. End

5. End
6. Illustrate the designed routing system using graphic plots
    a. Relative location of towns and hubs and indicate their connections
    b. Plot the evolution of cost against the generations.

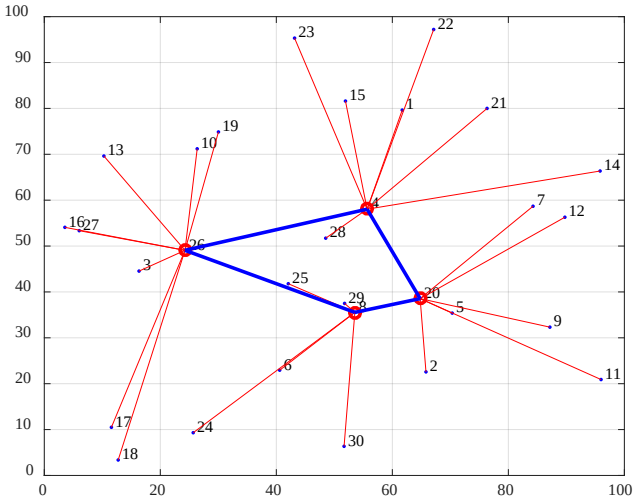

Figure 3. Example result of route planning.

In your program, you have to provide a plot of the minimum objective function against iterations and the resultant routes. Discuss the variation in the minimum objective function with respect to the number of iterations and the number of agents.

Assessment

This exercise carries 10% of the overall marks for the course.

The work is assessed according to the following criteria:

Absent	0%
Work less than 1/4 completed	2%
Work less than 1/2 completed	4%
Work more than 1/2 completed	6%
Work more than 3/4 completed	8%
Work all completed	10%

At the end of the scheduled laboratory period (week 11), the work (code and graphs) has to be marked-off by tutors. A signed copy (by tutors) of the mark sheet (download from Moodle) should be kept by the student and upload to Moodle within the week when marks are obtained. **No report is required for this exercise**.

Note: Time has to be allowed for tutors to mark the exercises. Students are expected to register with the tutor for their order in the marking process.