All tests

ALL\_test\_16t6m();

ALL\_test\_17t5m();

GA

* source code file: GA.m
* All test files match GA\_\*
* example
* iterations = 50;
* m = 3;
* n = 6;
* J = [2, 3, 4, 6, 2, 2];
* [costs, bestSol] = GA(J, m, iterations);

PSO

* source code file: PSO.m PSO\_lbest.m (better performance)
* All test files match PSO\_\*
* example
* iterations = 50;
* m = 3;
* n = 6;
* J = [2, 3, 4, 6, 2, 2];
* [costs, bestSol] = PSO\_lbest(J, m, n, 1500, iterations, @cost);

**Problem Description**

Job shop scheduling is an optimization problem in which n jobs J1, J2, …, Jn of varying sizes are given. These jobs need to be scheduled on m identical machines, while trying to minimize the makespan. The makespan is the total length of the schedule (that is, when all the jobs have finished processing).

**Problem Formulation**

**Variables**

* n, number of Jobs
* m, number of identical Machine
* J, an array of each Jobs’ weight
* S, an array of each Jobs’ Schedule

**Constraints**

* m > 1
* n > m
* ∀ s ∈ S, 1 <= s <= m

**Cost Function**

Time takes the longest scheduled machine to finish. See cost.m

**Goals**

* Minimize cost function
* Minimize number of iterations for each algorithm
* Find the best algorithm for the problem

**Simple Example**

* J = (2,3,4,6,2,2)
* S = (1,2,2,3,1,1)
* Cost = 7
* This setup is optimal

**Genetic Algorithm**

**Overview**

This part uses Genetic Algorithm to find the optimal solution for the job scheduling problem. The process was inspired by the evolution of organisms in natural. It employs random crossover, mutation and evolution to achieve the goal of finding the optimal scheduling for a set of given jobs. This process is based on the stock Genetic Algorithm given by the professor.

**Initial state**

* The population size is set to 100
* Chromosome length depends on the range of the possible output
* Crossover Probability was set to 95%
* Mutation probability was set to 5%
* There will be 2 sites of mutation, when the mutation event occurs

**Crossover**

* The crossover will exchange chromosome information at a specified crossover site, which is generated randomly.
* After each crossover, evolve will be called, and the fittest of the older population, or its offspring will survive.

**Evolve**

* The evolve function will maximize the model function, 1/(1+cost), which is the same as to minimize the cost
* The old and the new population will be compared, and the fitter of the two will get passed to the next generation

**Mutate**

* A given number mutation sites were generated, and the binary bits at the generated mutation sites will be flipped
* Evolve function will be called, and the older generation and the newer generation will be compared, the fittest of the two will get passed on to the next generation

**Particle Swarm Optimization**

**Overview**

This part uses the Ring Topology or lbest Particle Swarm Algorithm to find optimal solution for job scheduling problem. Each particle is communicating with four of its adjecent neighour. In each iteration, each particle calculates its speed based on the best solution in its neighbour and its personal best. Speed and location is defined in n dimensions.

**Initial state**

* All particals starts with 0 speed at all n directions.
* All particals starts at location randomly assigned between 1 ~ m in all dimensions.
* Local best solution is the same as partical’s location
* Neighbor best solution in each particle is the best solution in four of its neighours based on neighbor index.

**Local search criteria**

* Speed is calculated based on each particle’s personal best solution and the best solution of its neighbor. c1 = 1.4944, c2 = 1.4944, w = 0.9, vt+1i = w× vti+c1r1i(pbestti-xti)+ c2r2i(Nbestti-xti)
* The new solution is calculated by adding its previous location and its new speed, xt+1i = xti+vt+1i
* When the new cost of the new location is smaller than a particle’s local best, it updates its local best and update its neighbour’s neibour best when applicable.
* Asynchronous update method is used to reduce run time load requirement, neighbor best is updated when all partical finishes its calculation for its current round.

**Termination Criteria**

* The algorithm is terminated when set number of particals completes set number of iterations.
* The number of particals determines the amount of exploration and the amount of iterations determines the amount of exploitation.