University of Waterloo
Department of Electrical and Computer Engineering
Spring 2014 – ECE457A Cooperative and Adaptive Algorithms

CPAMI INTELLIGENT RESEARCH EXCELLENCE

IEEE Computational Intelligence Society

10. Solving Job Shop Scheduling Problem using Metaheuristics Algorithms

Job Shop Scheduling

The job shop scheduling problem (JSSP) is an optimization problem which can be applied in real workshops. The main target is to assign a certain number of jobs (n) to a certain number of machines (m) so that to achieve the minimum makespan. Usually the number of jobs is larger than the number of machines and different jobs may have different processing time, so different assignment strategies will lead to different makespan. There are m^n assignment options in total, so it is not viable to illustrate all possible assignment options for large m and n. In this case, how to use algorithms to achieve optimum objectives or acceptable sub-optimum objectives in a certain amount of time is a challenging work. Our motivation to attempt the JSSP is because it is a challenging problem and it is highly related to real life.

Problem Statement &Modeling

The variables in this project are defined as follows:

- n, number of Jobs
- m, number of identical Machine
- J, an array of each Jobs' weight (processing time)
- S, an array of each Jobs' Schedule

Constraints:

$$n > m > 1, \qquad n, m \in Z^+$$

$$\forall s \in S, \qquad s \in Z^+ \& 1 \le s \le m$$

Cost function:

$$f(x) = max_i f(i), \qquad i = 1,2,\dots,m$$

Where $f(i)$ is the makespan on i-th machine

Our objective is to minimize the cost function subject to these constraints.

For example:

J=(2,3,4,6,2,2)

S=(1,2,2,3,1,1) is an optimal solution

The optimal cost is 7

Metaheuristics

Tabu Search

Initial Solution
The initial solution is randomly generated.
Neighbours of each schedule
Each schedule will have (m-1)*n neighbours, where m is the number of machines, and n is the number of jobs. neighbours will only have one job scheduled on a different machine.

Finding the best neighbour
In order the find the neighbour with the lowest cost, the algorithm will loop through every valid neighbour and evaluate its cost. The neighbour with the lowest cost will be selected as the best neighbour.
Tabu List
The list length of the tabu list is user-defined. The tabu list acts like a queue (first in first out) in which the oldest move will be deleted when a new move is appended. A new move is appended every time after finding a best neighbour.

Simulated Annealing

Initial solution
Randomly generate an initial solution
Generating new solution
Using swap to generate new solutions (inserting & deleting is not necessary in this problem)
Update strategy
If the new solution is better than best so far solution, accept

Otherwise, calculate the difference between these two objective values Δf

Calculating $p = e^{-\Delta f/T}$, and generate a random number r

If p>r then accept, otherwise decline

Cooling schedule
Set the initial temperature T_0=30, and using the geometric cooling schedule to update the temperate at every two iterations

$$T_{new} = \partial \times T_{old}$$

Set $\partial = 0.85$
Set the final temperature to be 0.01

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.
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

PSO

Overview
This part uses the Ring Topology or lbest Particle Swarm Algorithm to find optimal solution for job scheduling problem. Each particle communicates with four adjacent neighbour. Each particle calculates its speed based on the best solution in its neighbour and its personal best.
Initial state
- All particles starts with 0 speed at all n directions.
- All particles starts at location randomly assigned between 1~m in all dimensions, Local best solution is the same as particle's location

Local search criteria
- Speed is calculated based on each particle's personal best solution and the best solution of its neighbour. c1 = 1.4944, c2 = 1.4944, w = 0.9
- The new solution is calculated by adding its previous location and its new speed, - 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 neighbour best when applicable.
Asynchronous update method is used to reduce run time load requirement, neighbour best is updated when all particle finishes its calculation for its current round.
Termination Criteria
The algorithm is terminated when set number of particles completes set number of iterations.

ACO

Overview
This part uses Ant Colony System to find the optimal solution for the job scheduling problem. The process is similar to find a shortest path between two nodes on a weighted tree graph.
Initial state
All ants starts at layer 0 of the tree, which means no job has been scheduled.
All nodes have initial pheromone of 1.
Pheromone will decrease 40% after each round.

Local search criteria
Local search depends on the number of pheromone, and the cost to move the next level. The cost is calculate by the extra number of time required for including the next job in certain machine. The cost can be zero. Experience VS Explore the new scheduling is used. A random value is generate to compare with r0. If the rand value is smaller than r0, the local search will select the route with max amount of pheromone.

Otherwise, it will do a roulette wheel selection based on $\frac{pheromone}{route\ cost+1}$

Pheromone deposit
Only the global best ants can deposit pheromone on their path.
The number of pheromone deposited equals to $\Delta\tau = \frac{1}{best\ cost}$.

Experimental Results

Experiment Setup

Two dataset generated by brute force algorithm, and all metaheuristics algorithm tested against the two dataset. Each metaheuristics algorithm runs 1000 iterations for each dataset.

Evaluation Metrics

1. Result (40%)
2. Total CPU time for 1000 iterations (20%)
3. Convergence CPU time for each algorithm (40%)

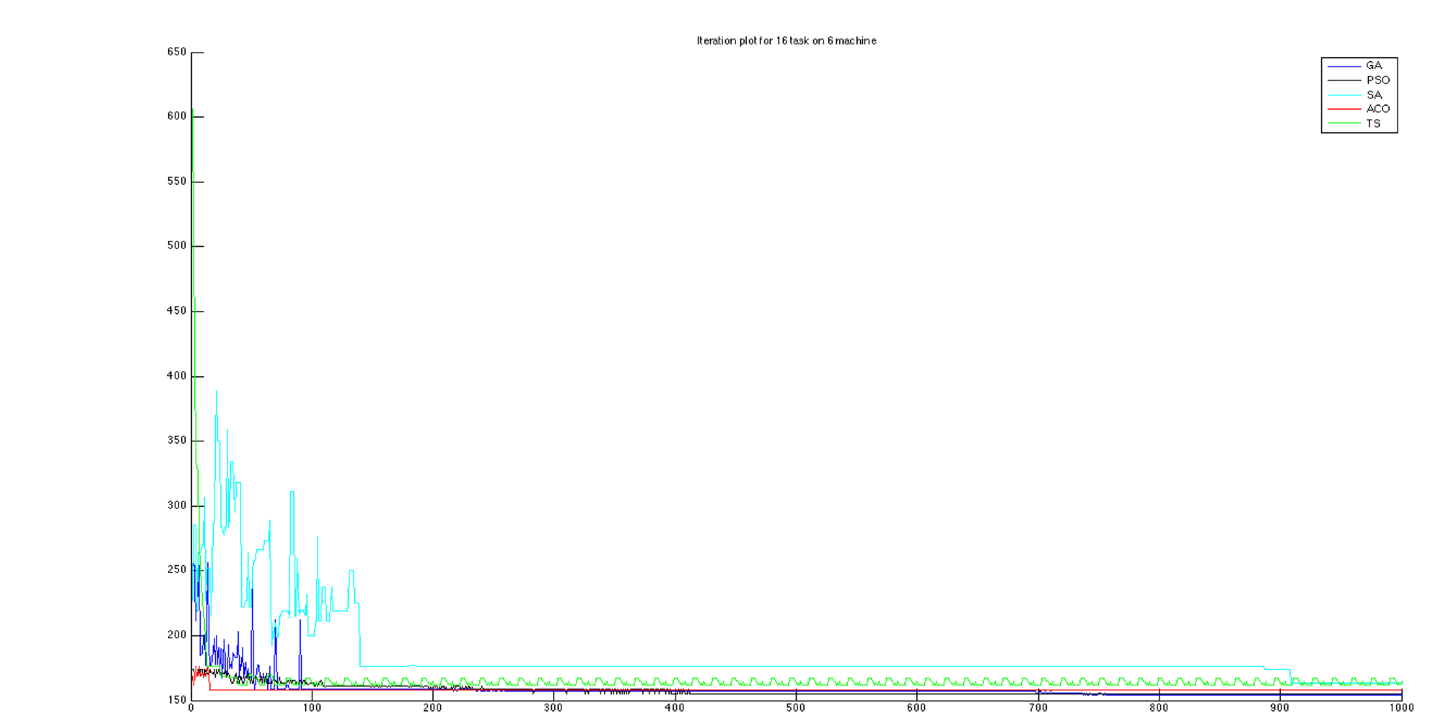
Comparative Study

Total time for 1000 iterations

data set	GA	PSO	TS	SA	ACO
16t6m	23.15	6.490	6.677	0.062	49.23
17t5m	22.62	6.209	5.600	0.047	44.15

Time of Convergence

data set	GA	PSO	TS	SA	ACO
16t6m	16.97	2.726	0.133	0.002	0.985
17t5m	9.048	5.588	0.168	0.011	0.883



Conclusion

Based on our experiment of 5 different metaheuristics algorithm for two different test points of our simplified job shop scheduling problem, it is concluded that in the same number of iterations GA finds the most optimal solution, and SA takes the minimum CPU time to converge. In the same number of iterations, TS finds the least optimal solution, while consuming a moderate amount of CPU time to converge, therefore it is considered the worst solution for this problem. SA finds the second worst solution however it takes only a fraction of CPU time comparing to other algorithms, therefor SA is still a good solution for this problem when computation resources are limited and only a sub-optimal solution is desired. For both test points, ACO finds the third best solution however it consumed large amount of CPU time to converge, therefore it's not considered a desired solution. PSO finds the second best solution and also takes second most CPU time to converge. GA finds the most optimal solution and take the most CPU time to converge. Therefore depending on the amount of CPU power and the how good the solution needs to be, PSO and GA are both good solution for this problem.

Yuanqi Li, Libin Bai, Rui Zhao, Debin Li, and Bolun Yao