Research Journal of Applied Sciences, Engineering and Technology 4(4): 383-386, 2012
ISSN: 2040-7467

Submitted: October 31, 2011 Accepted: November 25, 2011 Published: February 15, 2012

# Optimal Robot Arm Movement using Tabu Search Algorithm

S. Mobaieen, A. Rabii and B. Mohamady
Department of Engineering, Abhar Branch, Islamic Azad University, Abhar, Iran

**Abstract:** This study presents an optimum approach to calculate the optimal robot arm movement for processing a considerable commitment of tasks using Tabu Search (TS) algorithm. In the scheduling problem, the objective is to minimize the total processing time related to tasks distances from each other. In the first step, the TS method is reviewed and we employ the proposed method in order to assign efficiently the optimal robot arm movement. In our proposed algorithm, the crossover rate is large at first and gradually it is decreased based on convergence improvement in next generations. We define an objective function including the operation times. Then, by minimizing this function using discrete TS algorithm, the optimal robot arm movement trajectory is assigned efficiently and quickly. If the resulted best cost converges to global minima, the crossover rate will be decreased in next generation. This method is studied in terms of operation time, convergence speed and quality of the results. Superior features of this algorithm are fast tuning, rapid convergence, less computational burden and capability to avoid from local minima. High promising results demonstrate that our proposed method is very efficient and can obtain higher quality solutions with better computational capability.

**Key words:** Arm movement, genetic algorithm, particle swarm optimization, reduced crossover, tabu search, task-sequencing

## INTRODUCTION

The problems of applying mathematical approaches on engineering and optimization difficulties have led to the development of different heuristic solutions. Traditional procedures reach local optimum solutions and fail in solving hard problems with non-linear cost functions and various variables (Haupt and Haupt, 2004). Traditional methods such as mathematical programming and branch and bound methods are very slow and inefficient to solve this problem (Bryan and Alidaee, 2002). To overcome these complications, the favorite aim is to recommend a competent algorithm that may appear near-optimal solutions. Evolutionary algorithms are stochastic techniques that reproduce the natural evolutions. These processes have been established to reach near-optimal results of optimization problems which previous mathematical devices may fail on solving them.

The robot task-scheduling problem can be observed as a routing problem with finite number of cities that must be visited by a salesman person. The number of feasible solutions is ($n$!) (Bean, 1994). This problem is an NP-hard problem settled in Traveling Salesman Problem (TSP). The salesman starts the route from a point and visits all of the cities in a certain time interval. The cities in the TSP correspond to the tasks and operation times are related to distances of the cities (Bryan and Alidaee, 2002; Bean, 1994). All tasks are assumed to be available for processing at time zero and the goal is to minimize the whole operation time.

Genetic Algorithms (GAs) are the first, main and most popular technique of evolutionary techniques that was developed based on the Darwinian Principle, "survival of the fittest", and the biological process of evolution through reproduction (Al-Tabtabai and Alex, 1999). Based on the capability of GA method to attain near-optimal solutions, it has been used in many science applications. The first algorithm was developed by Holland (1975), and popularized by Goldberg (1989) who was able to use it for applied applications such as the control of gas-pipeline transmission (Al-Tabtabai and Alex, 1999; Goldberg, 1989). GA as a powerful adaptive search technique and stochastic search method for solving optimization problems is the most extensively known evolutionary algorithm. GA has been used in several optimization problems comprising job-shop and machine scheduling problems. Particle Swarm Optimization (PSO) was formulated by Kennedy and Eberhart (1995). In PSO algorithm, each solution to the problem is considered as a bird flying in the search space and is called a particle (Kennedy and Eberhart, 1995).

In this study, we present an optimum approach to calculate the optimal robot arm movement for processing a considerable commitment of tasks using TS. In the proposed method, the crossover rate is large at first and gradually it is decreased based on convergence improvement in next generations. An objective function including the operation times is defined and the efficiency of three intelligent algorithms, i.e., GA, PSO and TS is compared. In an attempt to improve the superiority of

**Corresponding Author:** A. Rabii, Department of Engineering Abhar Branch, Islamic Azad University, Abhar, Iran, Tel.: +982166285831

solutions and avoid trapping in local optima, we introduce TS method to the cost improvement for robot task-scheduling problem.

## METHODOLOGY

**Problem description:** A set of *n* tasks is presented to an intelligent robot for processing in particular time. The objective is to calculate the optimal robot arm movement to minimize the total operation time. The cost function to be minimized is:

$$T = \sum_{k=1}^{n} P_k + \sum_{k=0}^{n-1} d_{k(k+1)} + d_{n0} \quad (1)$$

where $T$ is the total operation time of robot on $n$ tasks. $d_{k(k+1)}=d_{ij}$ if the jobs $i$ and $j$ are considered in the $(k)^{th}$ and $(k+1)^{th}$ positions of the sequence. $P_i$ is the operation time on task $i$ and $d_{ij}$ is considered as follows:

$$d_{ij} = \max\left(r_{ij}, v_{ij}\right) \quad (2)$$

where $r_{ij}$ is the preparation time of robot for operation on task $j$ after task $i$, and $v_{ij}$ is the robot arm movement time from position of task $i$ to the position of task $j$. In Eq. (1), $d_{oj}$ is the movement time from the start position of the robot arm to the first task's position and $d_{jo}$ is the movement time from the position of the last task to the start position. Supposing that all processing times are fixed, the cost function may be reduced to:

$$T = \sum_{k=0}^{n-1} d_{k(k+1)} + d_{n0} \quad (3)$$

**An overview of tabu search algorithm:**

**GA:** The GA method is based on the mechanics of natural genetics and natural selection, and uses the following basic operators to manage all genetically activities of population individuals (Al-Tabtabai and Alex, 1999; Goldberg, 1989): selection or reproduction, which allows the best solutions to be transferred to the next generation, crossover, a random exchange of parents' genes, and mutation, a random change in the characteristics of a gene. Initial GA methods were presented in forms of Binary-coded sequences. Afterwards, new versions of GA were innovated, which used integer amounts to state sequences. In GA method, Firstly the preliminary random and corresponding task sequences are formed and algorithm calculates all cost values. The created solutions are selected for reproduction and crossover rate is assigned based on best cost rate and its closeness to optimal value. Then, crossover and mutation operations are performed. At the end of each generation, the stop criterion is checked. If the number of generations reaches the maximum, records the latest global best solution and ends the algorithm.

**PSO:** Considering the social behavior of swarm of fish, bees and other animals, the concept of the PSO is developed. The PSO is a robust stochastic evolutionary computation method based on the movement of swarms looking for the most fertile feeding location (Kennedy and Eberhart, 1995; Liu *et al.*, 2011). In general, PSO implementation is easier than GA. Indeed, PSO only has one operator; velocity calculation, so the computation time is decreased significantly. The reason is PSO does not perform the selection and crossover operations in evolutionary process.

Another difference between GA and PSO is the ability to control convergence. Crossover and mutation rates can affect the convergence of GA, but nothing can compare to the level of control achieved through manipulating of the inertial weight. The more decrease of inertial weight the more increase the swarm's convergence. This type of control allows determining the rate of convergence, and the level of "stagnation" eventually achieved. Stagnation occurs in GA when all of the individuals have the same genetic code. In that case the gene pool is uniform, crossover has little or no effect on population and each successive generation is essentially same as the first. However, in the PSO, this effect can be controlled or prevented.

All solutions in PSO can be represented as particles in a swarm. Each particle has a position and velocity vector and each position coordinate represents a parameter value. Similar to the most optimization techniques, PSO requires a fitness evaluation function relevant to the particle's position.

**TS:** TS algorithm first was introduced by (Glover, 1990; Goldberg, 1989; Padmavathi and Shalinie, 2010). This approach is a combination of optimization and search methods based on converging to the best available neighborhood solution point. In other words, the principle idea is to move to the best point of search space in terms of cost function, even though it is worse than the current solution point (Padmavathi and Shalinie, 2010; Porto *et al.*, 2000). An aspiration criterion implies that if some tabu movements lead to promising solutions, these tabu status are accepted. The TS method is a local search approach with a flexible memory structure and it can be applied directly to various problems, without requiring the plant models.

TS procedure is utilized with features that lead to escape from local minima. Important features consist of Short-Term Memory, Long-Term (Frequency-Based) Memory and Aspiration Criteria. In the Short-term Memory (Tabu List) we maintain a list of solution points that have been visited recently and must be avoided. We update the tabu list in each iteration based on solution points.

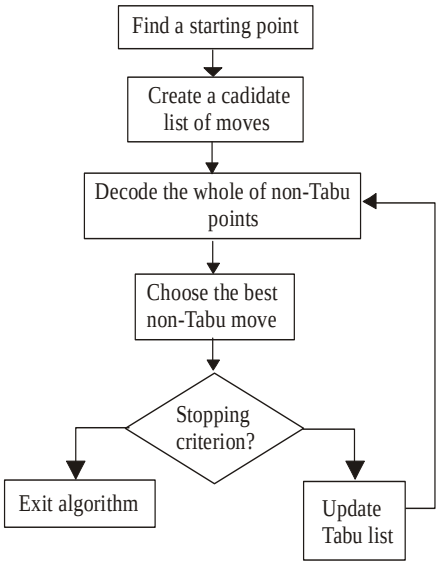


Fig. 1: Flow chart of TS algorithm

In the Long-Term Memory we register the number of occasions that the solution point has been visited. The updating period of this memory is very longer than short-term memory. We use this memory while all of the non-tabu points in the neighborhood lead to increase the cost function and aspiration criterion is not satisfied.
Aspiration Criteria implies that if some tabu moves lead to promising solutions, these moves are accepted.

In this section we summarize the discrete TS algorithm in five steps. Assume that X is a total search space and x is a solution point sample and f(x) is cost function. First, we choose $x \epsilon X$ to start the process, then a candidate list of non-Tabu moves in neighborhood is created. All solution points are decoded and then we find $x_{winner} \epsilon N(x)$ such that $f(x_{winner}) < f(x_i), i \neq Winner$ . In the last stage, we check the stopping criterion. If satisfied, exit the algorithm. If not, $x = x_{winner}$ , the Tabu List is updated and we return to the first stage of algorithm.

There are several criterions for exiting the algorithm:

- Determining a threshold $T_0$: If the value of cost function is less than $T_0$, stop algorithm
- Determine specific number of iterations
- If the value of the cost is remained invariable or negligible change for several iterations, exit the algorithm

Flow chart of the algorithm is shown in Fig. 1. In this paper, we use discrete Tabu Search algorithm for task scheduling applications as well.

## SIMULATION RESULTS

Considering the random sequences, the order of tasks are assigned so that if we start with the smallest number and place its position number at first position of the task sequence, the second task will be relative to the next smallest number and so on until all tasks' positions are assigned. Using these random sequences, it is easy to perform crossover on them. Also, immigration operation can be added to the algorithm by producing a new random sequence instead of the worst sequence (instead of the parent with worst cost).

At first step, parents' elements are generated between 0 and 1 randomly and task sequences are assigned according to these random generated numbers (Table 1). Then, cost function for each sequence is calculated and the program sorts all random and task sequences based on their corresponding cost values (Table 2). Random and corresponding task sequences in the second generation are illustrated in Table 3.

The best solution is allowed to forward to the next generation and the crossover is performed on two sequences which have low costs (sequences 1 and 2 or 2 and 3) proportional to cost convergence to the global

Table 1: Random generated sequences and their related task sequences.

| Random sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.7393 | 0.0150 | 0.4093 | 0.1960 | 0.9342 | 0.0409 | 0.3685 | 0.9515 | |
| 2 | 0.3162 | 0.5816 | 0.5147 | 0.2861 | 0.6802 | 0.6948 | 0.1546 | 0.6072 | |
| 3 | 0.8734 | 0.1411 | 0.5559 | 0.2965 | 0.2788 | 0.4749 | 0.0120 | 0.2269 | |
| 4 | 0.3770 | 0.7201 | 0.5753 | 0.7459 | 0.5800 | 0.9050 | 0.3949 | 0.4435 | |
| 5 | 0.4544 | 0.0337 | 0.3276 | 0.5928 | 0.5533 | 0.0460 | 0.1150 | 0.0391 | |
| Task sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Cost |
| 1 | 6 | 1 | 5 | 3 | 7 | 2 | 4 | 8 | 184.9814 |
| 2 | 3 | 5 | 4 | 2 | 7 | 8 | 1 | 6 | 185.6138 |
| 3 | 8 | 2 | 7 | 5 | 4 | 6 | 1 | 3 | 150.5182 |
| 4 | 1 | 6 | 4 | 7 | 5 | 8 | 2 | 3 | 144.0916 |
| 5 | 6 | 1 | 5 | 8 | 7 | 3 | 4 | 2 | 187.0537 |

Table 2: Sorted task sequences based on their cost values.

| Task sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Cost |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 6 | 4 | 7 | 5 | 8 | 2 | 3 | 144.0916 |
| 2 | 8 | 2 | 7 | 5 | 4 | 6 | 1 | 3 | 150.5182 |
| 3 | 6 | 1 | 5 | 3 | 7 | 2 | 4 | 8 | 184.9814 |
| 4 | 3 | 5 | 4 | 2 | 7 | 8 | 1 | 6 | 185.6138 |
| 5 | 6 | 1 | 5 | 8 | 7 | 3 | 4 | 2 | 187.0537 |

Table 3: Random and corresponding task sequences in the second generation (crossover rate is 5/8).

| Random sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.3770 | 0.7201 | 0.5753 | 0.7459 | 0.5800 | 0.9050 | 0.3949 | 0.4435 | |
| 2 | 0.8734 | 0.7201 | 0.5753 | 0.2965 | 0.5800 | 0.9050 | 0.012 | 0.4435 | |
| 3 | 0.3770 | 0.015 | 0.5753 | 0.7459 | 0.9342 | 0.0409 | 0.3949 | 0.4435 | |
| 4 | 0.4322 | 0.0541 | 0.7885 | 0.0167 | 0.9506 | 0.9320 | 0.0239 | 0.5466 | |
| 5 | 0.9764 | 0.9143 | 0.6017 | 0.1655 | 0.4636 | 0.8969 | 0.2020 | 0.2604 | |
| Task sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Cost |
| 1 | 1 | 6 | 4 | 7 | 5 | 8 | 2 | 3 | 144.0916 |
| 2 | 7 | 6 | 4 | 2 | 5 | 8 | 1 | 3 | 173.4596 |
| 3 | 3 | 1 | 6 | 7 | 8 | 2 | 4 | 5 | 154.3209 |
| 4 | 4 | 3 | 6 | 1 | 8 | 7 | 2 | 5 | 131.8701 |
| 5 | 8 | 7 | 5 | 1 | 4 | 6 | 2 | 3 | 139.9540 |

Table 4: Best cost comparison of algorithms

| | Best cost | | | |
|---|---|---|---|---|
| Jobs | NJS | GA | PSO | TS |
| 8 | 134.021 | 92.7586 | 92.7586 | 89.4853 |
| 12 | 194.361 | 171.9197 | 163.3117 | 160.1921 |
| 20 | 404.578 | 356.904 | 331.0989 | 330.3287 |
| 50 | 1204.2 | 1174.7 | 1106.11 | 1101.6548 |

Table 5: Run time comparison

| | Run time (s) | | | |
|---|---|---|---|---|
| Jobs | NJS | GA | PSO | TS |
| 8 | 0.033064 | 1.901204 | 0.4844 | 2.4839 |
| 12 | 0.021937 | 2.6592 | 0.46912 | 2.9795 |
| 20 | 0.1342 | 4.3173 | 0.55975 | 5.4387 |
| 50 | 0.0945 | 22.882 | 2.2434 | 24.1658 |

minima. For example, at first generations, because of much difference between the best cost and the near-optimum value, the crossover rate is large and therefore majority of sequences elements are exchanged. In next generations, by approaching the cost value to minima, crossover is performed on few numbers of elements. In the following, we compare Neighborhood Job Search (NJS) (Bryan and Alidaee, 2002), GA, PSO and TS methods.

Table 4 and 5 show the results in forms of the best cost and run time for different numbers of jobs.

It is clear from the simulation results in Table 4 and 5 that although the run time for task sequencing using the TS algorithm is more than the other methods, however it has reasonable improvement in terms of the cost comparison. This competence is illustrated better by increasing the number of jobs and confronting with more complicated problems.

## CONCLUSION

In this study, we present the Tabu Search algorithm for the task sequencing problem. A new step has been introduced to the framework, crossover rate reducing based on the cost value improvement. Compared with other methods, this proposed algorithm demonstrates the capability of producing better solutions though taking longer time. The topic of our future researches is combining our mentioned algorithm with the PSO method to reduce the computation time and improve the algorithm efficiency.

## REFERENCES

Al-Tabtabai, H. and P. Alex, 1999. Using genetic algorithms to solve optimization problems in construction. Eng. Constra. Archit. Manage., 6(2): 121-132.

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comp., 6(2): 154-160.

Bryan, R. and B. Alidaee, 2002. Single machine scheduling to minimize total weighted late work: A comparison of scheduling rules and search algorithms. Comp. Indus. Engineering, 43: 509-528.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Addison-Wesley Publishing Co, MA.

Glover, F., 1990. Tabu Search-Part II. ORSA J. Comp., 2(1): 4-32.

Haupt, R.L. and S.E. Haupt, 2004. Practical Genetic Algorithms. State College, Pennsylvania, a John Wiley & Song, Inc.,

Holland, J., 1975. Adaptation In Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Kennedy, J. and R.C. Eberhart, 1995. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, IEEE Sevice Center, pp: 1942-1948.

Liu, L.L., G.P. Zhao and S.S. Ou'Yang, 2011. Integrating theory of constraints and particle swarm optimization in order planning and scheduling for machine tool production. Inter. J. Adv. Manufactur. Technol., 57(4): 285-296.

Padmavathi, S. and S.M. Shalinie, 2010. Tabu search based memetic algorithm for task scheduling. J. Comp. Inf. Syst., 6(4): 1017-1025.

Porto, S.C.S., J.P. Kitajima and C.C. Ribeiro, 2000. Performance evaluation of a parallel tabu search task scheduling algorithm. Parallel Comp., 26(1): 73-90.