Qiuyun, Tao, et al. “Improved particle swarm optimization algorithm for AGV path planning”

Son, Minwoo

Operation Research Lab.

2023-08-18

Contents

1 Introduction

2 Problem Description

3 Proposed Algorithm

4 Conclusion

Introduction

1 Introduction

2 Problem Description

3 Proposed Algorithm

4 Conclusion

Paper Summary

This paper. . .

1. Deals with Automated Guided Vehicle (AGV) path-planning problem of a one-line production line in a workshop
2. Establishes mathematical model to minimise the transportation time, then proposing an improved particle swarm optimisation (IPSO)
3. Presents a new coding method: a crossover operation to update the particle position of PSO to avoid falling into a local optimum
4. Shows the efficiency and effectiveness of the newly developed algorithm in material transportation.

Background

Problems regarding AGVs are classified into two categories:

1. Task scheduling problem (Job assignment)
    - Using AGV in a manufacturing workshop enviroment contributes to solving scheduling problem with acquring the best solution
2. Path planning problem
    - Checking out feasibility of a path between two points
    - Obtaining conflict-free/deadlock-free path
    - Planned path to be optimised for the efficiency of the entire workshop

Background

Problems regarding AGVs are classified into two categories:

1. Task scheduling problem (Job assignment)
    - Using AGV in a manufacturing workshop enviroment contributes to solving scheduling problem with acquring the best solution
2. ***Path planning problem***
    - Checking out feasibility of a path between two points
    - Obtaining conflict-free/deadlock-free path
    - ***Planned path to be optimised for the efficiency of the entire workshop***

Background

PSO Algorithm

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 (p_{id}^{k} - x_{id}^{k}) + c_2 r_2 (p_{gd}^{k} - x_{id}^{k})$$
$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}$$

Various methods using PSO algorithm have been developed to solve scheduling problems:

- PSO with local search strtegy to solve single machine scheduling problem, Li *et al.* (2019)
- PSO with human learning optimisation for flexible job scheduling problem, Ding & Gu (2020)

Idea from Previous Study

Previous research mostly used pure dynamic programming, genetic algorithm, heuristic algorithm and etc on path planning problem.

Especially, basic PSO algorithm has shortcomings as follows:

1. Only suitable for continuous problems
2. Not appropriate to deal with combinatorial problems
3. Easy to fall into a local optimality

Idea from Previous Study

To solve the problems mentioned earlier, this paper incorporates into PSO such additional methods as:

1. Integer coding method to make PSO suitable for path planning problem
2. Crossover operations to update particle positions
3. Mutation mechanism to have particles escape from local optimalities

Problem Description

1 Introduction

2 Problem Description

3 Proposed Algorithm

4 Conclusion

Problem Description

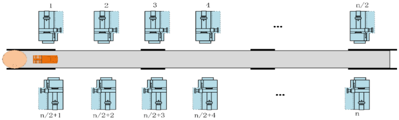


FIGURE 1. Schematic diagram of one-line workshop production line.

$$F = \min \left\{ \sum_{i=1}^{m} t_{i-1,i} \right\} \qquad (1)$$

$t_{i-1,i}$: the time between the $i-1$ th task and the ith task
m: the number of machines requesting materials

other constraints are left out

- One-line production
- Trasporting materials to machine tools with **the shortest time**
- With n machines in total, half of them are situated at the bottom and top respectively
- The number of machines(m) requesting materials can be less than $n(m \leq n)$.

Proposed Algorithm

1 Introduction

2 Problem Description

3 Proposed Algorithm

4 Conclusion

Methodology

Feature of Improved Particle Swarm Optimisation (IPSO)

- Encoding of Particles
- Initialisation of Particle
- Crossover Operation
- Mutation Operation

Above are methods proposed to resolve issues discussed in Introduction

Methodology

1. **Encoding of Particles**

- An AGV-related problem is a discrete optimisation problem.

  → The data needs to be coded with discrete values.
- A particle of PSO is represented with a vector of *integers*.

  e.g., $X_i = (3,1,0,5,9,8,6,7,2,4)$ *for* $i \leq$ (# of iterations)

–	0	1	2	3	4	5	6	…
no	9	2	7	11	18	8	10	…
time	0	120	200	250	320	500	650	…

no is the No. of a machine tool

time is the time when a machine calls for material.

2. **Initialisation of Particle**

This is not specific to this algorithm but a general process.

1. Determine particle length based on the number of machines requesting.
2. Generate random numbers for parameters of PSO for each m machines.
3. Confirm all machines tools are included in the initial vector.
4. Repeat 100 times to create 100 initial particles.

Methodology

3. **Crossover Operation**

By setting crossover probability $G = 1$, every particles are updated with the crossover operation.

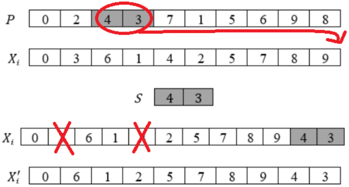


FIGURE 2. **Cross operation of particle.**

Step of the operation

1. Randomly chooose the segment $S = S_1, S_2$ from local or global optimal solution.
2. Insert chosen segment from step 1 into the particle X_i. 3
3. Delete S_1, S_2 from particle X_i.

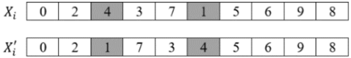
Methodology

4. Mutation Operation

Mutation operation is introduced to *avoid falling into a local optimum* and to *prevent early convergence*.

X_i	0	2	4	3	7	1	5	6	9	8
X_i'	0	2	7	4	3	1	5	6	9	8

FIGURE 3. **Insertion operation.**



X_i	0	2	4	3	7	1	5	6	9	8
X_i'	0	2	1	7	3	4	5	6	9	8

FIGURE 4. **Reverse sequence mutation.**

- **Insertion operation**
  Randomly choose an element in the particle X_i and insert it into another position.
- **Reverse sequence mutation**
  Randomly choose two elements in the particle X_i and then swap the positions of them.

Methodology

Procedure of Algorithm

1. Initialise all particles randomly. (The population size is 100)
2. Calculate objective values and then save the values and the optimal solution of each group.
3. Perform crossover operation.
4. Perform mutation operation with probability $Q = 0.2$.
5. Verify the optimality of the newly generated solution: update the gloabl best solution if needed.

Experiment: Setting

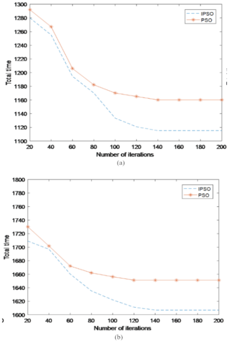
- The results are compared with PSO (without mutation), genetic algorithm (GA) and ant colony optimisation (ACO).
- Two cases are experimented to compare simple and complex situations:
  1. Case 1: *10* machine tools calling material
  2. Case 2: *25* machine tools calling material
- Criterion to compare results: $\frac{F-F_b}{F_b} \times 100\%$; the lower, the better

F_b is the shortest time among those by all algorithms.

F is the average time of each algorithm after 25 experiments.

Experiment: Result Analysis

Comparison 1: with PSO



(a)

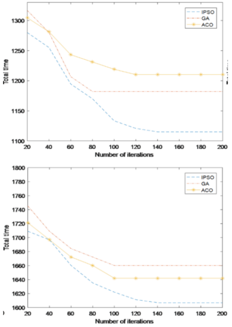
(b)

	Case 1	Case 2	Mean
IPSO	**3.43**	**3.16**	**3.30**
PSO	8.27	5.37	6.82

- PSO converges prematurely and thus fails to find an efficient solution.
- IPSO produces better solutions in both cases than basic PSO.
- IPSO has stronger search ability with escpaing from a local optimum and avoiding premature convergence.

Experiment: Result Analysis

Comparison 2: with GA and ACO

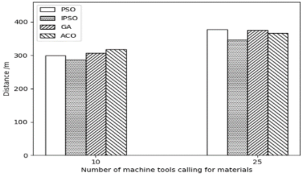


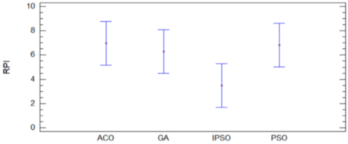
	Case 1	Case 2	Mean
IPSO	**3.43**	**3.16**	**3.30**
GA	7.19	4.89	6.04
ACO	8.68	5.27	6.98

- Both GA and ACO converge prematurely compared to IPSO.
- IPSO produces better solutions in both cases than GA and ACO.
- Observing converges rates of each algorithm, IPSO proves to be effective in jumping out of a local optimum.

Experiment: Result Analysis

Additional Analysis





- With IPSO, the distance traveled by AGV also is minimised.
- The criterion of IPSO is statistically significantly better than other algorithms.
- IPSO shows stability in acquring best solutions.

Conclusion

1 Introduction

2 Problem Description

3 Proposed Algorithm

4 Conclusion

Contribution

Minimising traveling time of a single AGV enviroment for mutiple machines and **path optimisation of the AGV** have been effectively successful through IPSO.

Modifications are as follows:

- A new coding method for solving AGV path planning problem with PSO is proposed
- Particle positions are updated based on crossover operation
- Mutation operation is applied to escape from local optimum and enhance efficiency of local search

Additional Thought

Applicability

- Incorporating conventional ideas from other methods to solve a certain problem (from GA into PSO)
- Checking the efficiency of IPSO in a more complex enviroment
    1. Mutiple working AGVs
    2. Restrictions on areas to travel
- Not just mere path-planning but also job sequencing in a way

Additional Thought

Doubtful Points

- Computationally efficient?
- Meaningfully better results?
- Appropriate measures to prove the relevance of the expeirment?

Thank You!