Int J Adv Manuf Technol (2013) 64:399–409
DOI 10.1007/s00170-012-3999-1

ORIGINAL ARTICLE

An ant colony-based algorithm for finding the shortest bidirectional path for automated guided vehicles in a block layout

Mahdi Hamzheei · Reza Zanjirani Farahani · Hannaneh Rashidi-Bajgan

Received: 1 August 2010 / Accepted: 13 February 2012 / Published online: 8 March 2012

Abstract This paper considers the shortest path design problem (SPDP) on bidirectional path topology as one of the best known types of network configurations for automated guided vehicles. An integer linear programming model has been developed to solve the problem. The model intends to minimize the length of the path, which needs to cover all cells at least in one edge. Due to the NP-hardness of the problem, which has been proved previously, this model is only able to solve problems with a small number of cells. So we develop an ant colony system (ACS) algorithm to solve the problem. Comparisons of the designed algorithm with a cutting-plane algorithm show the efficiency of the proposed ACS algorithm for this SPDP.

Keywords Bidirectional path · Automated guided vehicle · Block layout · Ant colony system · Branch-and-cut method

M. Hamzheei
Department of Industrial and System Engineering,
University of Wisconsin,
Madison, WI, USA

R. Z. Farahani (✉)
Department of Informatics and Operations Management,
Kingston Business School, Kingston University,
London, UK
e-mail: zanjiranireza@gmail.com

H. Rashidi-Bajgan
Young Researchers Club, Karaj Branch,
Islamic Azad University,
Alborz, Iran

1 Introduction

As one of the most important activities in industrial engineering, facility planning consists of two main parts: facility layout and facility location. Facility layout deals with decisions regarding the departments' layout, production units, manufacturing units, storage places, and so on. Furthermore, facility layout is defined in terms of facility system design, material handling system design, and layout design [62]. Based on the estimation made by Tompkins et al. [62], about 20% to 50% of production costs are related to layout design and material handling. Therefore, any cost reduction in these fields can potentially result in significant overall cost savings.

To have a good layout design, it is crucial to integrate material handling decisions in layout design [2]. Therefore, planning and analyzing in the layout is barely possible without considering the material handling system. The material handling system is defined by Tanchoco and Sinriech [61] based on material handling equipment, configuration and direction of material handling networks, and the number and locations of pick-up and delivery (P/D) stations. In this paper, we consider automated guided vehicles (AGVs) as material handling equipment among the other types.

Regarding the subject of the problem, there are many issues that could be included in the designing and controlling of the AGV system such as guide–path design, vehicle routing and scheduling, deadlock resolution, the number of vehicles, idle-vehicle positions, loading/unloading positions, battery management, etc. [27, 41, 67]. These issues belong to different levels of the decision-making processes, like strategic level (guide–path design), technical level (scheduling), and operational level (vehicle routing). Whereas each subject stands alone, there are some relations between different issues. This paper focuses on shortest path design problem (SPDP) which is counted as a

strategic decision in material handling system design. Since this is a static design, some managing attributes like the order of queue, AGV distribution strategy, and avoidance of collisions are not provided in this already complex decision-making problem, which lies in the basic stage of layout design. We concentrate on designing the shortest guide–path serving all cells of the manufacturing floor on the block layout.

1.1 Contribution

In this section, we describe what we are exactly looking for as a guide path on the block topology. A block layout is a set of cells, which are represented by rectangles but not necessarily convex polygons, packed beside each other. A path is defined feasible if it satisfies the two following conditions:

1. The path must be adjacent to each cell at least in one edge.
2. The path must be non-intersecting, i.e., the path should not cross itself. In this regard, Figs. 1 and 2 show infeasible and feasible paths, respectively.

If one of these conditions is violated, then the path will be infeasible. There might be no feasible path in a block layout (e.g., when a cell enclaves another cell). Given the block layout, the objective function of our algorithm in this paper is to minimize the length of the feasible path quickly.

1.2 Application

Our algorithm is an economical decision-making method that could be applied in designing the layout of flexible manufacturing floors as well as transportation designs. Moreover, SPDP generally could be useful in networking and creating integrated circuits. In addition, since for a non-deterministic polynomial time (NP)-hard combinatorial optimization problem, the mathematic model is common, such as SPDP, traveling salesman problem (TSP), job–shop problem, vehicle routing problem (VRP),

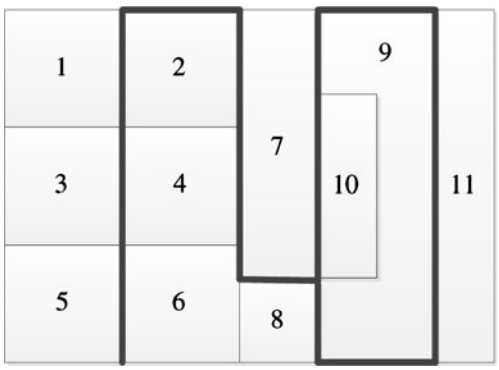

Fig. 1 A block layout with an intersected (infeasible) path

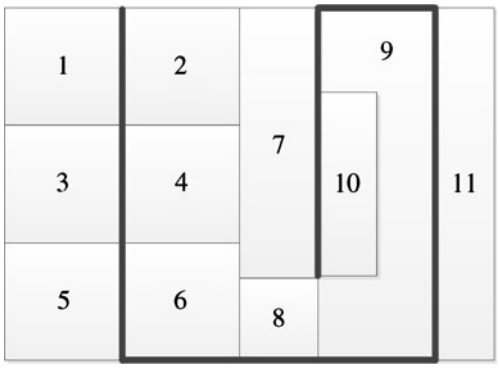

Fig. 2 A block layout with feasible path [20]

etc., so our methodology for dealing with SPDP could be applicable for solving mathematical models of these problems, too.

1.3 Motivation

For the first time, this problem has been studied by Maxwell and Muckstadt [44]. They find and install the best routes for AGVs, and then in order to efficiently transfer materials in a block layout, the maximum number of required vehicles is determined. This problem has been followed by a large number of researchers and scholars, and varied heuristic and meta-heuristic methods have been developed in this research theme [7, 17, 36, 54, 61]. Here, we are trying to track and fix the time-consuming hole of our previous problem-solving method while optimality and efficiency are considered. Previously, the integer linear programming (ILP) model introduced by Hamzeei and Farahani [28] was solved by using a cutting-plane algorithm. This algorithm as an exact method is not able to solve instances of this NP-hard problem with large sizes in a reasonable amount of time. So, we develop an ant colony system (ACS) algorithm, which, as a construction-based meta-heuristic method, is efficient enough in solving different sizes of this problem.

In fact, (a) in the literature, we show that the efficiency of such algorithm has not previously been investigated to solve such problems (probably because it is newer than other meta-heuristics), and (b) the nature of this algorithm, which is from the routing skill of ants, makes it easier to adjust with these problems.

1.4 Research questions

In this research, the following questions are answered:

- Is there any other efficient way to solve the supposed SPDP on the block layout?
- Does the ACS algorithm support optimality and save time in solving the problem?

- Why and how is ACS more appropriate among other heuristic and meta-heuristic methods for the concept of SPDP?

1.5 Organization

The paper is organized as follows. Section 2 explains related works to our problem. The ACS algorithm is explained in Section 3.Computational results are presented in Section 4. Finally, Section 5 contains the conclusion and future research roads. The model of Hamzeei and Farahani [28] is explained in the Appendix.

2 Related works

Researchers have studied various configurations for the material handling network so far. These configurations include conventional configuration, unidirectional single loop (USL), tandem configuration, segmented flow topology (SFT), and bidirectional path.

Gaskins and Tanchoco [25] have formulated a model for the conventional configuration. Thereafter, mathematical programming has been widely used for conventional configuration [19, 26, 30, 31, 36, 37, 55, 57, 60, 63].

USL finds a loop serving all P/D stations in a block layout which has been studied by several researchers [4–8, 17, 18, 20–22, 39, 56, 61]. Tandem configuration designs multiple loops with transfer stations; this model has been usually used in cellular manufacturing systems [10, 11, 23, 24, 32, 35, 40, 42, 51]. SFT partitions the layout into non-overlapping segments, and each segment is served by a single vehicle [3, 9, 58].

Another type of material handling network is bidirectional path with two opposite directions to flow [12, 29, 33, 45, 47]. In the current paper, we consider the bidirectional shortest path as the material handling network configuration.

Afentakis [1] expresses the advantages of the loop pattern for network configuration; these are simplicity and efficiency, low initial and expansion cost, and flexibility in process and production. In addition, there are some advantages of bidirectional path configuration which are counted as encouraging factors for its use rather than single loop configuration. For example, the total traveled distance by vehicles is reduced in bidirectional paths [16]. Subsequently, this improves the response time of the vehicles [64], and from the economical point of view, a bidirectional path needs less space rather than a single loop. Kim and Tanchoco [33, 34] show in a particular network that the bidirectional layout outperforms the unidirectional one in terms of the number of completed jobs per unit time. Asef-Vaziri et al. [5] state that sometimes it is impossible to imply a single loop in a special block layout; however, the bidirectional path could be implemented more easily at the same layout. High productivity is another benefit of bidirectional paths [41].

Egbelu and Tanchoco [16] assert that different alternatives could be assumed in the adjustment of a bidirectional path system: (a) parallel tracks on each aisle, (b) single switchable track, and (c) mixed design. Despite the fact that flow of traffic can take place in either direction of each aisle in a bidirectional guide–path system, vehicles are not permissible to travel in opposite direction at the same time. Thus, there should be buffers for temporary parking areas. They discuss about different types of buffers and the locations and number of required buffers, and then introduce a model which describes the flow and control of AGVs in a bidirectional network.

Among the different research papers, there are two related works, which inspired us to develop this paper. Rajagopalan et al. [48] use Lagrangian relaxation to address material flow path and the location of P/D stations simultaneously. They consider two models on the bidirectional flow path; the first model supposes unloaded vehicles, and the second one supposes loaded and unloaded vehicles for a collision-free path. Hamzeei and Farahani [28] discuss two ILP models in a cutting-plane approach to find the shortest path in a block layout.

A summary of related works on bidirectional path is shown in Table 1. The symbols used in the columns of this table are explained below.

1. Objective function could be:
   - OF1: The shortest path
   - OF2: Minimizing the total travel distances of the loaded vehicles
   - OF3: Minimizing the total travel distances of the loaded and unloaded vehicles
   - OF4: Minimizing cost of the network (fixed cost)
2. Input data could be:
   - I1: The layout
   - I2: The from–to chart
3. Output result could be:
   - O1: The path
   - O2: The locations of the P/D stations

As it is clear in the illustration of Table 1, branch-and-cut, Lagrangian relaxation, Petri net, and more recently some meta-heuristic methods such as genetic algorithm (GA) and simulated annealing (SA) have got a lot of interests from researchers in this field. These papers show that heuristic and meta-heuristic algorithms could outperform well in facing different bidirectional shortest path problems.

Table 1 Summary of related works on the bidirectional path

References	Objective function	Inputs	Solving technique	Outputs	Others
Egbelu and Tanchoco [16]	OF1	I1	Simulation	O1	Up to 12 cells solved
Kim and Tanchoco [33]	OF1	I1	Time windows based on Dijkstra algorithm	O1	Conflict-free path up to 6 cells solved
Chhajed et al. [12]	OF1, OF4	I1, I2	Lagrangian relaxation, two-phase heuristic includes building and improvement phases	O1	Up to 12 cells solved
Kim and Tanchoco [34]	OF1	I1	Column generation	O1	Conflict-free path up to 9 cells and 5 vehicles simulated
Krishnamurthy et al. [38]	OF2	I1	Column generation-based heuristic	O1	Conflict-free path up to 18 cells solved
Rajotia et al. [49]	OF3, OF4	I2	Heuristic methodology	O2	Up to 6 cells and 14 vehicles solved
Ventura and Lee [64]	OF1	I1	Exact polynomial-time algorithm	O1, O2	Up to 300 stations and 30 AGVs considered
Wu and Zhou [65]	OF3	I1	Colored resource-oriented Petri net modeling method	O2	Collision and deadlock-free path simulated
Rajagopalan et al. [48]	OF2, OF3	I1, I2	Lagrangian relaxation	O1, O2	Up to 15 cells solved
Sarker and Gurav [53]	OF2	I1	Heuristic routing algorithm	O1	Conflict-free path simulated
Hamzeei and Farahani [28]	OF1	I1	Branch-and-cut approach	O1	Up to 45 cells solved
Wu and Zhou [66]	OF1	I1	Petri net method	O1	Deadlock and blocking avoidance simulated
Srivastava et al. [59]	OF3	I1	Intelligent agent-based AGV controller	O2	Collision and deadlock-free path simulated
Ren et al. [50]	OF2	I1	Shortest time routing algorithm based on best-first search for Petri net	O1	Avoiding deadlock, blocking, and collision considering time window by using state graph
Liu [43]	OF2	I1	Heuristic based on GA	O1	Deadlock-free control simulated

It could be said that ACS as a well-known meta-heuristic has been neglected among them, while it seems ACS has a suitable methodology in finding paths. In the following section, we invoke the methodology of ACS and pick it among other meta-heuristic methods to show it is extremely adaptable and efficient in finding the shortest path quickly.

This paper is going to consider ACS to solve the problem because (a) the efficiency of this technique has not been considered in solving such problems and (b) the nature of ACS which is from the routing of ants makes it easier to implement in such problems.

3 Ant colony system algorithm

This section shows how we develop an ACS algorithm to solve SPDP. First, we briefly explain the ACS technique. Then, we define the sets, indices, and parameters used in our algorithm.

3.1 ACS technique

Dorigo et al. [15] developed ant colony optimization (ACO) based on the nature of ants in seeking foods. Ants are able to find the shortest path between a food source and their nest and vice versa by smelling a chemical substance called pheromone. While walking, ants leave pheromone on the ground, and each of them chooses to follow a direction probabilistically based on the amount of pheromone on that path. This is the main reason of using ants' ability to find the shortest path in optimization.

The first ACO algorithm was implemented for the TSP. Afterwards, VRP, quadratic assignment problem as combinatorial optimization problems were solved by ACO algorithms. Interested readers are referred to a more complete list of these applications showed at http://www.aco-methaheuristic.org. It was shown that ACO algorithms are one of the best existing algorithms used for routing problems [46].

For implementing an ACO algorithm, a group of virtual ants search the space of the problem to reach the best possible answer. By depositing pheromone on the edges of the graph, each ant communicates with other ants that walked through the space, and in this way, the efforts of ants are synergized. After a while, as the process continues, the amount of pheromone on the shortest path will be more than the other paths. Eventually, the edge which has been mostly visited is chosen as a feasible solution. Dorigo and Stützle [14] have published a prime book which includes a number of different ACO algorithms. In this paper, we use an algorithm called ACS proposed by Dorigo and Gambardella [13].

3.2 The algorithm

In this section, indices, parameters, and variables used in the algorithm are presented, and then, the ACS algorithm is defined.

3.2.1 Sets and indices

- $C=\{1, 2, \ldots, n\}$: The set of cells in block layout graph, $(r, t \in C)$
- $V=\{1, 2, \ldots, v\}$: The set of nodes in block layout graph, $(i, j \in V)$
- $E=\{1, 2, \ldots, e\}$: The set of edges in block layout graph, which is defined as $E=\{(i, j) \in V \mid i, j \in V$ are adjacent nodes$\}$
- $Mj=\{i \in V \mid (i, j) \in E$ or $(j, i) \in E$: The set of all nodes adjacent to node j
- P^c: The complement set of P, $(P^c=V \backslash P)$

3.2.2 Variables and parameters

- y_i^k: The binary variable which is equal to 1 if and only if node i is adjacent to the path for ant k; otherwise $y_i^k = 0$
- x_t^k: The binary variable which is equal to 1 if and only if cell t is adjacent to the path for ant k; otherwise $x_t^k = 0$
- c_{ij}: The length of edge (i, j)
- S: The index of starting node of the path
- f: The index of finishing node of the path

According to the above variables, we now define the following sets:

- $P^k = \left\{ i \in V \middle| y_i^k = 1 \right\}$: The set of all nodes forming a certain path for ant k
- $N^k = \left\{ i \in N \cup \{0\} \middle| x_t^k = 0 \right\}$: The set of all cells not adjacent to the path for ant k

3.2.3 Procedure of generating a feasible path

In this part of the algorithm, we introduce a procedure for generating a feasible path in the block layout. It is important to note that each edge (i, j) is the intersection of two cells in the block layout including $\{0\}$ representing the area containing the block layout. In this procedure, a_1 and a_2 denote two cells which intersect at (r, r'). In our ACS algorithm, each ant uses this procedure to find a feasible path. The procedure is as follows:

Step 1. Let $N=C$. Generate r between 1 and n. Select the r th member of N, and then let $P=\{r\}$ and $s=r$.

Step 2. Select r' using transition rule from $M_s \cap N^k$. Let $P=P \cup \{r'\}$ and $f=r'$.

Step 3. Consider the cells a_1 and a_2 adjacent to edge (r, r'), then let $N \leftarrow N \backslash \{a_1, a_2\}$. If $|N| = 0$, stop; otherwise if $|M_s \cup M_f| \neq 0$ then go to step 4. If $|M_s \cup M_f| = 0$, go to step 1.

Step 4. Select r' using the transition rule from set $\{M_s \cup M_f\} \cap P^c$. Let $P \leftarrow P \cup \{r'\}$. If $(r', s) \in E$, let $r=s$ and $s=r'$. If $(r', f) \in E$, let $r=f$ and $f=r'$. Now go to step 3.

3.2.4 Implementation of ACS for solving SPDP

In this section, in order to solve the SPDP, we present an ACS algorithm. First, m dummy ants are randomly positioned on m vertices of a given block layout. Then, ACS is applied.

ACS uses state transition rule, local updating pheromone rules, and global updating pheromone rules. We determine the following approaches to implement these rules.

At each stage of constructing a feasible path, the stage transition rule is applied by the dummy ant indexed with k to choose the new vertex i. The stage transition rule is described in Eq. 1.

$$S_i^k \begin{cases} 1 \quad , & \text{if } q \geq q_0 \text{ and } i = \operatorname{argmax}_{j \in J_k} \left\{ [\tau_j]^{\alpha} \left[\eta_j\right]^{\beta} \right\} \\ \dfrac{[\tau_i]^{\alpha}[\eta_i]\beta}{\sum_{j \in M_f \cap N^k} [\tau_i]^{\alpha}[\eta_i]^{\beta}} \quad , & \text{if } q \geq q_0 \\ 0 \quad , & \text{otherwise} \end{cases} \tag{1}$$

where S_i^k is the probability of selecting node i as the next node to form a feasible path for ant k. In addition, τ_i is the amount of pheromone deposited on edges adjacent to node i. Moreover, η_i is equal to $\frac{1}{\sum_{(i,j) \in E} C_{ij}}$, the inverse value of the sum of all edges adjacent to node i. α and β are two parameters which determine the relative effect of pheromone signs (τ_i) and heuristic information (η_i), respectively. $M_t \cap N^k$ is the set of acceptable nodes for node i of ant k.

When ant k decides to select a node to form a feasible path, a random number q will be generated from the interval (0, 1). If q is less than a specified value q_0, the node with the maximal value of $[\tau_j]^{\alpha}$ $[\eta_j]^{\beta}$ is selected from set $M_t \cap N^k$; otherwise, according to the probability which is given in the second part of the transition rule in Eq. 1, the next node is selected from set $M_t \cap N^k$

After forming a feasible path for each ant, the local updating rule on the nodes of the path is applied in order to update the pheromone and prevent pheromone concentration on a specific path. The pheromone on vertex i is modified by local updating rule with the following equation:

$$\tau_i = (1 - \phi)\tau_i + \phi\tau_0 \tag{2}$$

where $\phi \in (0, 1)$ is the pheromone trail evaporation rate, and τ_0 is the initial value of the pheromone deposited on each node, which is equal for all nodes.

After forming a feasible path by all ants, the global updating rule will update the amount of pheromone on each node. The rule is given by the following relations:

$$\tau_i = (1 - \rho)\tau_i + \rho \Delta\tau_i \tag{3}$$

$$\Delta\tau_i = \begin{cases} \frac{1}{L_b} & , i \in P^k \\ 0 & , \text{otherwise} \end{cases} \tag{4}$$

where $\rho \in (0, 1)$ is the pheromone decay parameter, and L_b is the length of the best path so far found. Indeed, this rule with an appropriate value of ρ rewards the nodes on the shortest path.

In order to implement the ACS algorithm, the following pseudo-code is applied:

Select the iteration number $t = 0$.
Calculate η_i for $i = 1, 2, \ldots |V|$.
Repeat
  Select the ant number $k = 0$.
  Repeat
    Run *feasible path generating procedure* for ant *k*.
    Calculate the length of new path.
    If the length of new path is shorter than the best found path, then replace the best path with the new path.
    Apply *local updating rule.*
      $k \leftarrow k + 1$
  Until *k* is less than *ant number*.
  Apply *global updating rule.*
Until *t* is less than iteration number.

4 Computational results

In order to show the efficiency of our algorithm against the cutting-plane algorithm by Hamzeei and Farahani [28], we compare these two. The cutting-plane algorithm has been solved using LINGO 8.00 [52], and the ACS algorithm is coded using Microsoft Visual Basic 6.0. Both algorithms were run on a PC with Pentium 4, 1,700 MHz CPU and 256 MB of RAM. For this paper, 15-, 20-, 25-, 30-, 35-, 40-, and 45-cell test instances are used. Since there are no known test instances for our problem in the literature, for each size, five instances were randomly generated ending up with total number of 35 instances.

4.1 The parameters of the algorithm

The parameters of the algorithm including iteration number, ant number, q_0, α, β, ρ, ϕ and τ_0, and τ_0 should be set to the certain values. We consider iteration number, ant number and τ_0 as coefficients of the cell number, so:

$$\text{iteration number} = a_0 \times \text{cell number} \tag{5}$$

$$\text{ant number} = \left\lceil \frac{\text{cell number}}{b_0} \right\rceil \tag{6}$$

$$\tau_0 = \frac{\text{cell number}}{c_0} \tag{7}$$

For a_0, b_0, and c_0, we assumed three levels (10, 15, 20), (3, 4, 5), and (0.6, 0.7, 0.8), respectively. Furthermore, for each of the other parameters, three levels are considered: (0.3, 0.4, 0.5) for q_0, (1,2,3) for α,(1,2,3) for β, (0.2, 0.3, 0.4) for ρ, and (0.1, 0.11, 0.12) for ϕ.

To determine the parameters, three different large-enough problems whose sizes are 35, 40, and 45 are given. The parameters are not decided simultaneously; first, we determine the value of a_0, then the best value of b_0 is decided, and so on. Actually, we solve the three chosen problems for determining the value of a_0 while the other undecided parameters are fixed to their middle values (e.g., ϕ is fixed to 0.11). For each problem, the algorithm is run with three values (10, 15, 20) for a_0. Among all nine results, one with the least average deviation from the optimal value is chosen. Once the value of a_0 is fixed, we start finding the value of b_0 in the same process we proceed for a_0, and this is continued for determining all our parameters. Finally, for a_0, b_0, c_0, q_0, α, β, ρ, and ϕ, the determined values are 15, 0.7, 5, 0.4, 2, 1, 0.04, and 0.12, respectively.

4.2 Analysis of the results

Having set the value of all the parameters, the test instances are solved by both cutting-plane and ACS algorithms whose results are shown in Table 2. In Table 2, the results of cutting-plane algorithm are presented under the column titled "Cutting-plane algorithm," which includes two sub-columns such as average time of finding optimal solution and the number of optimum solutions found in five test instances. For ACS algorithm, the "Average computational time" shows the average time that the algorithm needs to terminate. This column shows the efficiency of the proposed algorithm timewise.

The comparison of the average computational time of ACS and cutting plane is shown in Fig. 3 implying that ACS is remarkably faster than cutting plane on average.

In order to show the quality of solutions of the ACS algorithm, we compute the deviation from optimum for each test instance:

a The objective function value of the cutting-plane algorithm

b The objective function value of the ACS

$$\text{Deviation from optimum} = \frac{b-a}{c} \times 100 \tag{8}$$

Then, we compute the average of these values for each size, and the results are shown under "Average deviation from optimum" in Table 2. The maximum average is 3.09% for instance size of 45.

Another the index for showing the quality of solutions of the ACS algorithm is the "Maximum deviation from optimum" for each size. In this column, the maximum value is associated with size 25 which is 7.14%. Therefore, the maximum deviation for all of the test instances is 7.14%.

Finally, we show the number of test instances for which the ACS could find the optimal value. Among 35 test instances used for testing the ACS algorithm, 21 test instances, 60% of the test instances, could be solved to optimality by the ACS algorithm.

5 Conclusion

This paper considers the SPDP problem in which an ACS algorithm is developed to solve the problem. We pursue the ILP presented by Hamzeei and Farahani [28] and compare the optimality of our algorithm with the previously developed cutting-plane algorithm. In this regard, 35 test problems are solved. The computational results show that in terms of time and quality of the solutions both, the ACS could solve small-sized problems efficiently well and acts for large problems reasonably well. The computational times of the ACS algorithm are significantly less than the cutting-plane algorithm for each size. The average deviations from optimum, maximum deviation from optimum, and number of problems solved to optimality show the quality of the developed ACS.

Table 2 The computational results of cutting plane and ACS algorithms

No. of cells	No. of test instances	Cutting-plane algorithm		ACS algorithm			
		Average time of finding optimal solution (s)	No. of optimum solutions	Average computational time (s)	Average deviation from optimum (%)	Maximum deviation from optimum (%)	No. of optimum solutions
15	5	3.4	5	1.2	0.00	0.00	5
20	5	27.4	5	3.2	0.00	0.00	5
25	5	143.4	5	5.6	2.81	7.14	2
30	5	425	5	11.8	0.78	2.00	3
35	5	905	5	41.4	2.34	4.62	2
40	5	2,278	5	27.2	0.91	3.03	3
45	5	5,935	5	53.8	3.09	6.04	1

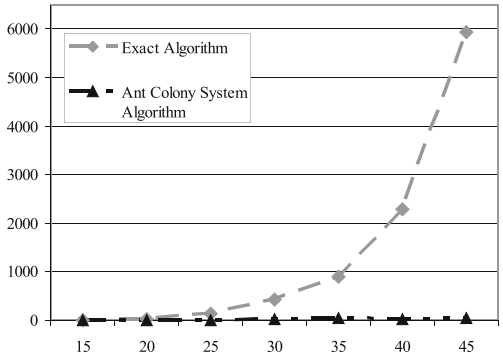

Fig. 3 Average computational time for both algorithms

There are several points for future researches as follows:

- To improve ACS, some modification in setting parameters could be done. Based on the characteristic of ACS, a small value of pheromone evaporation would lead to a premature convergence to a sub-optimal value especially in large search spaces. The use of a high value of it especially in the early generations of the search decreases the possibility of premature convergence. This might indicate why our ACS algorithm performance degraded with a larger number of cells in Table 2. Therefore, using a suitable design of experiment framework like the Taguchi method and supposing a proper range of values for parameter setting could improve the performance of the ACS algorithm.
- Since the developed ACS is a quite quick algorithm to find the suitable shortest path, this could be applicable in dynamic management of the AGV system, which for sure would bring more assumptions of managing level like distributing, routing and scheduling vehicles, and avoiding collisions [43, 66]. These assumptions make the problem more complex and need a large investigation in comparing with a variety of heuristic and meta-heuristic methods.
- Designing the block layout and finding an optimal bidirectional path simultaneously is a strategy that can end up more saving comparing to this hierarchical strategy.

Acknowledgments The authors would like to thank the anonymous referees and also the editor for their valuable comments and suggestions that helped them to improve the paper.

Appendix: Basic algorithm

In our "Computational results" section, we compared the result of our ACS algorithm with that of Hamzeei and Farahani [28]. Here, we briefly outline their algorithm.

First, the sets and indices of their algorithm are shown. Afterwards, the variables and parameters are introduced. Finally, their ILP model and branch-and-cut algorithm will be described.

Sets and indices

- $C = \{1, 2, \ldots, n\}$: The set of cells in a block layout graph ($p \in C$)
- $S \subset C$: The subset of C
- $\overline{S} \subset C/S$: The complement set of set S
- $V = \{1, 2, \ldots, v\}$: The set of nodes in block layout graph ($i, j, k, l \in V$)
- $V_p = \{1, 2, \ldots, v_p\}$: The set of nodes of cell p in block layout graph, $\left(\forall p \in C, \bigcup_{p \in C} V_p = V\right)$
- $E = \{1, 2, \ldots, e\}$: The set of nodes in block layout graph, which is defined as $E = \{(i,j) \in V | i,j \in V$ are adjacent nodes$\}$
- $E_p = \{1, 2, \ldots, e_p\}$: The set of edges of cell p in block layout graph, $\left(\forall p \in C, \bigcup_{p \in C} E_p = E\right)$
- $E(S) = \{(i,j) \in E | a \in S : (i,j) \in a\}$: The set of all edges belonging to subset S in block layout graph

Now, we define an equation as follows:

$$A_{ab} = E_a \cap E_b \quad , \quad \forall a, b \in C \tag{9}$$

This equation defines a set which includes the common edges between two subsets of S. According to Eq. 1, the following subsets are defined:

- $S_A = \{S \subset C | \forall a \in S : \sum_{b \in S} |A_{ab}| \geq 1\}$: The set of all subsets of adjacent cells in block layout graph
- S_{A_m}: A member of S_A
- $B(S_A) = \{(i,j) \in E(S_A) | \exists b \in \overline{S}_A : (i,j) \in E_b\}$: The set of edges on boundary of S_A

Variables and parameters

- C_{kl}: The length of edge (k, l), $C_{kl} = \{(k,l) \in E | k,l \in V$ are adjacent nodes$\}$.
- x_{ij}: The binary variable which is equal to 1 if and only if (i, j) is adjacent to the path; otherwise, x_{ij}=0
- y_k: The binary variable which is equal to 1 if and only if node k is adjacent to the path; otherwise, y_k=0
- v_k: The binary variable which is equal to 1 if and only if node k is at the start or finish node of the path; otherwise, v_k=0

Mathematical model

The mathematical model supposed for solving SPDP could be found below:

$$\text{Min} \sum_{i<j} c_{ij} x_{ij} \tag{10}$$

subject to

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2y_k - v_k \quad , \quad \forall k \in V \tag{11}$$

$$\sum_{(i,j)\in E_p} x_{ij} \geq 1 \quad , \quad \forall p \in C \tag{12}$$

$$\sum_{(i,j)\in B(S_{A_m})} x_{ik} \leq |B(S_{A_m})|-1 \quad , \quad S_{A_m} \in S_A \tag{13}$$

$$\sum_{k=1} v_k = 2 \tag{14}$$

$$x_{i,j}, y_k, v_k \in \{0,1\}, \forall k \in V, \quad i,j \in E \tag{15}$$

Equation 10 is the objective function, which minimizes the total length of the path. Equation 11 is the degree constraint. This equation is very similar to the one used by Asef-Vaziri et al. [5] for degree constraint. This means that if a node is in the middle of the path, two adjacent edges of it must be in the path. If a node is the start or finish node of the path, then only one of its adjacent edges are in the path. If a node is not in the path, then none of its adjacent edges could be in the path.

Equation 12 is covering constraint. This relation is similar to the covering constraint in Asef-Vaziri et al. [5]. However, if SPDP assumes a path is feasible if it is adjacent in at least one node, we can use another constraint as follows:

Equation 13 is tour eliminator constraint. This equation has a similar role as connectivity constraint in Asef-Vaziri et al. [5].

Equation 14 states that the path should only have exactly two nodes as the start and finish nodes.

Equation 15 is integrality constraint.

Algorithm

The difficulty of solving this model is in Eq. 13 because its number will increase exponentially as the number of cells increases. Therefore, for solving this model, a branch-and-cut approach is used, and Eq. 13 is selected for cutting. The algorithm is as follows:

Step 1. Set c=1 as the iteration counter. Initialize a linear problem in which constraints (11), (12), (14), and (15) are introduced.

Step 2. Solve the model with LINGO 10.00. If the solution is feasible, stop.

Step 3. Find members of set S_A for violated constraints (13). Apply constraints (13) for these members. Set c=c+1. Go to step 2.

References

1. Afentakis P (1989) A loop layout design problem for flexible manufacturing systems. International Journal of Flexible Manufacturing Systems 1:175–196
2. Apple JM (1977) Plant layout and material handling. Wiley, New York
3. Asef-Vaziri A, Goetschalckx M (2008) Dual track and segmented single track bidirectional loop guide path layout for AGV systems. European Journal of Operational Research 186:972–989
4. Asef-Vaziri A, Laporte G (2005) Loop based facility planning and material handling. European Journal of Operational Research 164:1–11
5. Asef-Vaziri A, Laporte G, Sriskandarajah C (2000) The block layout shortest loop design problem. IIE Transactions 32:727–734
6. Asef-Vaziri A, Dessouky M, Sriskandarajah C (2001) A loop material flow system design for automated guided vehicles. International Journal of Flexible Manufacturing Systems 13:33–48
7. Asef-Vaziri A, Laporte G, Ortiz R (2007) Exact and heuristic procedures for the material handling circular flow path design problem. European Journal of Operational Research 176:707–726
8. Banerjee P, Zhou Y (1995) Facilities layout design optimization with single loop material flow path configuration. International Journal of Production Research 33:183–203
9. Barad M, Sinriech D (1998) A Petri net model for the operational design and analysis of segmented flow topology (SFT) AGV systems. International Journal of Production Research 36:1401–1425
10. Bozer YA, Srinivasan MM (1991) Tandem configurations for automated guided vehicle systems and the analysis of single vehicle loops. IIE Transactions 23:72–82
11. Bozer YA, Srinivasan MM (1992) Tandem AGV systems: a partitioning algorithm and performance comparison with conventional AGV systems. European Journal of Operational Research 63:173–191
12. Chhajed D, Montreuil B, Lowe TJ (1992) Flow network design for manufacturing systems layout. European Journal of Operational Research 57:145–161
13. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1:53–66
14. Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge
15. Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy. Technical Report, University of Milan, Italy
16. Egbelu PJ, Tanchoco JMA (1986) Potentials for bi-directional guide-path for automated guided vehicle based systems. International Journal of Production Research 24:1075–1097
17. Eshghi K, Kazemi M (2006) Ant colony algorithm for the shortest loop design problem. Computers & Industrial Engineering 50:358–366
18. Farahani RZ, Laporte G (2004) Two formulations for designing optimal single loop and the location of P/D stations. 3rd International Industrial Engineering Conference, Amirkabir University of Technology, Tehran, Iran, 1–14
19. Farahani RZ, Tari FG (2001) Optimal flow path designing of unidirectional AGV systems. International Journal of Engineering Science 12:31–44
20. Farahani RZ, Laporte G, Sharifyazdi M (2005) A practical exact algorithm for the shortest loop design problem in a block layout. International Journal of Production Research 43:1879–1887
21. Farahani RZ, Karimi B, Tamadon S (2007) Designing an efficient method for simultaneously determining the loop and the location of the P/D stations using genetic algorithm. International Journal of Production Research 45:1405–1427

22. Farahani RZ, Pourakbar M, Miandoabchi E (2007) Developing exact and Tabu search algorithms for simultaneously determining AGV loop and P/D stations in single loop systems. International Journal of Production Research 45:5199–5222
23. Farahani RZ, Laporte G, Miandoabchi E, Bina S (2008) Designing efficient methods for the tandem AGV network design problem using tabu search and genetic algorithm. Int J Adv Manuf Technol 36:996–1009
24. Fazlollahtabar H, Rezaie B, Kalantari H (2010) Mathematical programming approach to optimize material flow in an AGV-based flexible job shop manufacturing system with performance analysis. Int J Adv Manuf Technol 51:1149–1158
25. Gaskins RJ, Tanchoco JMA (1987) Flow path design for automated guided vehicle systems. International Journal of Production Research 25:667–676
26. Gaskins RJ, Tanchoco JMA, Taghaboni F (1989) Virtual flow paths for free-ranging automated guided vehicle systems. International Journal of Production Research 27:91–100
27. Guan X, Dai X (2009) Deadlock-free multi-attribute dispatching method for AGV systems. Int J Adv Manuf Technol 45:603–615
28. Hamzeei M, Farahani RZ (2007) Two optimal algorithms for finding bi-directional shortest path design problem in a block layout. Journal of Industrial Engineering International 3:24–34
29. Hsueh C (2010) A simulation study of a bi-directional load-exchangeable automated guided vehicle system. Computers & Industrial Engineering 58:594–601
30. Kaspi M, Tanchoco JMA (1990) Optimal flow path design of unidirectional AGV systems. International Journal of Production Research 28:1023–1030
31. Kaspi M, Kesselman U, Tanchoco JMA (2002) Optimal solution for the flow path design problem of a balanced unidirectional AGV system. International Journal of Production Research 40:389–401
32. Kim KS, Chung BD (2007) Design for a tandem AGV system with two-load AGVs. Computers & Industrial Engineering 53:247–251
33. Kim CW, Tanchoco JMA (1991) Conflict-free shortest-time bidirectional AGV routing. International Journal of Production Research 29:2377–2391
34. Kim CW, Tanchoco JMA (1993) Operational control of a bidirectional automated guided vehicle system. International Journal of Production Research 31:2123–2138
35. Kim K, Chung B, Jae M (2003) A design for a tandem AGVS with multi-load AGVs. Int J Adv Manuf Technol 22:744–752
36. Ko KC, Egbelu PJ (2003) Unidirectional AGV guide path network design: a heuristic algorithm. International Journal of Production Research 41:2325–2343
37. Kouvelis PW, Kim M (1992) Unidirectional loop network layout problem in automated manufacturing systems. Operation Research 40:533–550
38. Krishnamurthy NN, Batta R, Karwan MH (1993) Developing conflict-free routes for automated guided vehicles. Oper Res 41:1077–1090
39. Laporte G, Asef-Vaziri A, Sriskandarajah C (1996) Some applications of the generalized travelling salesman problem. J Oper Res Soc 47:1461–1467
40. Laporte G, Farahani RZ, Miandoabchi E (2006) Designing an efficient method for tandem AGV network design problem using tabu search. Applied Mathematics and Computation 183:1410–1421
41. Le-Anh T, De Koster MBM (2006) A review of design and control of automated guided vehicle systems. European Journal of Operational Research 171:1–23
42. Lin JT, Chang CCK, Liu W (1994) A load-routing problem in a tandem-configuration automated guided-vehicle system. International Journal of Production Research 32:411–427
43. Liu SN (2011) Modeling and optimization of integrated scheduling of automated warehouse system. Advanced Materials Research 230–232:35–39
44. Maxwell WL, Muckstadt JA (1982) Design of automatic guided vehicle systems. IIE Transactions 14:114–124
45. Maza S, Castagna P (2005) A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles. Computers in Industry 56:719–733
46. Mohan BC, Baskaran R (2012) A survey: ant colony optimization based recent research and implementation on several engineering domain. Expert Systems with Applications 39:4618–4627
47. Qiu L, Hsu W (2001) A bi-directional path layout for conflict-free routing of AGVs. International Journal of Production Research 39:2177–2195
48. Rajagopalan S, Heragu SS, Don Taylor G (2004) A Lagrangian relaxation approach to solving the integrated pick-up/drop-off point and AGV flow path design problem. Applied Mathematical Modeling 28:735–750
49. Rajotia S, Shanker K, Batra JL (1998) An heuristic for configuring a mixed uni/bidirectional flow path for an AGV system. International Journal of Production Research 36:1779–1799
50. Ren XL, Wen HY, Li H (2008) Routing algorithm for multiple AGVs with the undirected Petri net. Journal of Xidian University 35:517–522
51. Rezapour S, Zanjirani-Farahani R, Miandoabchi E (2011) A machine-to-loop assignment and layout design methodology for tandem AGV systems with single-load vehicles. International Journal of Production Research 49:3605–3633
52. Roe A (1997) User's guide for LINDO and LINGO, Windows version, Belmont, CA
53. Sarker BR, Gurav SS (2005) Route planning for automated guided vehicles in a manufacturing facility. International Journal of Production Research 43:4659–4683
54. Sedehi MS, Farahani RZ (2009) An integrated approach to determine the block layout, AGV flow path and the location of pick-up/delivery points in single-loop systems. International Journal of Production Research 47:3041–3061
55. Seo Y, Egbelu PJ (1995) Flexible guide path design for automated guided vehicle systems. International Journal of Production Research 33:1135–1156
56. Sinriech D, Tanchoco JMA (1992) The centroid projection method for locating pick-up and delivery stations in single-loop AGV systems. Journal of Manufacturing Systems 11:297–307
57. Sinriech D, Tanchoco JMA (1993) Solution methods for the mathematical models of single-loop AGV systems. International Journal of Production Research 31:705–725
58. Sinriech D, Tanchoco JMA (1994) SFT—segmented flow topology. In: Tanchoco JMA (ed) Material flow systems in manufacturing. Chapman & Hall, London, pp 219–238
59. Srivastava S, Choudhary A, Kumar S, Tiwari M (2008) Development of an intelligent agent-based AGV controller for a flexible manufacturing system. Int J Adv Manuf Technol 36:780–797
60. Sun XC, Tchernev N (1996) Impact of empty vehicle flow on optimal flow path design for unidirectional AGV systems. International Journal of Production Research 34:2827–2852

61. Tanchoco JMA, Sinriech D (1992) OSL-optimal single-loop guide paths for AGVS. International Journal of Production Research 30:665–681
62. Tompkins JA, White JA, Bozer YA, Tanchoco JMA (2010) Facilities planning, 4th edn. Wiley, New York
63. Venkataramanan MA, Wilson KA (1991) A branch-and-bound algorithm for flow-path design of automated guided vehicle systems. Naval Research Logistics 38:431–445
64. Ventura JA, Lee C (2003) Optimally locating multiple dwell points in a single loop guide path system. IIE Transactions 35:727–737
65. Wu N, Zhou M (2004) Modeling and deadlock control of automated guided vehicle systems. IEEE/ASME Transactions on Mechatronics 9:50–57
66. Wu N, Zhou M (2007) Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking. IEEE/ASME Transactions on Mechatronics 12:63–72
67. Yahyaei M, Jam J, Hosnavi R (2010) Controlling the navigation of automatic guided vehicle (AGV) using integrated fuzzy logic controller with programmable logic controller (IFLPLC)—stage 1. Int J Adv Manuf Technol 47:795–807