Int J Adv Manuf Technol (2008) 36:996–1009
DOI 10.1007/s00170-006-0909-4

ORIGINAL ARTICLE

Designing efficient methods for the tandem AGV network design problem using tabu search and genetic algorithm

**Reza Zanjirani Farahani · Gilbert Laporte ·
Elnaz Miandoabchi · Saman Bina**

Received: 20 March 2006 / Accepted: 6 December 2006 / Published online: 12 January 2007

Abstract A tandem AGV configuration connects all cells of a manufacturing facility/plant by means of non-overlapping, single-vehicle closed loops. Each loop has at least one additional P/D station, provided as an interface between adjacent loops. This study describes the development of tabu search and genetic algorithm procedures for designing tandem AGV systems. The objective is to minimize the maximum workload of the system. Both algorithms have mechanisms to prevent solutions with intersecting loops. The new algorithms and the partitioning algorithm presented by Bozer and Srinivasan are compared using randomly generated test problems. Results show that for large-scale problems, the partitioning algorithm often leads to infeasible configurations with crossed loops in spite of its shorter running time. However the newly developed algorithm avoids infeasible configurations and often yields better objective function values.

Keywords AGV · Tandem configuration · Tabu search · Genetic algorithm

R. Zanjirani Farahani (✉) · E. Miandoabchi · S. Bina
Department of Industrial Engineering,
Amirkabir University of Technology,
Tehran, Iran
e-mail: farahan@aut.ac.ir

G. Laporte
Canada Research Chair in Distribution Management and GERAD,
HEC Montréal,
3000 chemin de la Côte-Sainte-Catherine,
Montreal H3T 2A7, Canada

S. Bina
Supply Chain Management Research Group,
Tehran, Iran

1 Introduction

The design of handling systems is one of the most important decisions in facility design activities. Material handling operations make up to nearly 20 to 50% of the overall operational costs [1]. An automated guided vehicle (AGV) is a driverless vehicle used for the transportation of goods and materials within a production plant partitioned into cells (or departments), usually by following a wire guide path. One of the most important issues in designing AGV systems is the guide path design. The problem of guide path design for AGVs is not new. A number of algorithms for AGV guide path design have been developed over the past 20 years [2]. The AGV guide path configurations discussed in previous research include *conventional/traditional* (see [3–14]), *tandem* (see [15–19]), *single loop* (see [20–29]), *bi-directional shortest path* (see [30–32]) and *segmented flow* ([33–36]). Vis (2006) discusses literature related to design and control issues of AGV systems in manufacturing, distribution, transshipment and transportation systems [37].

The tandem configuration, which is the concern of this paper, was introduced by Bozer and Srinivasan [15, 16] and is based on the "divide-and-conquer" principle. A tandem configuration is obtained by partitioning all the workstations into single-vehicle, non-overlapping zones. Additional pick-up/delivery (P/D) points are provided between adjacent zones to serve as transfer points. This configuration offers some advantages such as the elimination of blocking and congestion, simplicity of control, and flexibility due to system modularity. It also has some disadvantages including the need for handling a load by two or more vehicles and thus longer load movement times, extra floor space and cost requirements, resulted by the use of additional P/D points and conveyors. Lin et al. [18]

proposed a two phase approach to route loads by multiple vehicles in different zones through the tandem configuration network.

Tandem paths were initially proposed by Bozer and Srinivasan [15, 16] who presented an analytical model to compute the workload of a single loop. They developed a heuristic method which partitioned the stations in loops [17]. Hsieh and Sha [38] proposed a design process for the concurrent design of machine layout and tandem routes. Liu and Chen [39] suggested a divided AGVS which is similar to a tandem AGVS in that there is one AGV in each divided zone. However, the difference is that the path of one zone is allowed to cross over the path of another. Aarab et al. [40] used hierarchical clustering and tabu search to determine tandem routes in a block layout. Yu and Egbelu [41] presented a heuristic partitioning algorithm for a tandem AGV system, based on the concept of variable path routing. Kim et al. [42] proposed an analytical model to design a tandem AGVS with multi-load AGVs. Their design procedure is somewhat similar to the one proposed by Bozer and Srinivasan. They use TSP to generate subsets of stations and then apply Markov's chain model to check their serviceability. Then, they select the final tandem path using a partitioning model. Using simulations, the performance of the proposed model is shown through comparing it with a conventional multi-load AGVS. Later Bozer and Lee [43] considered eliminating the conveyers by using an existing station as a transfer point. Ventura and Lee [44] studied tandem configurations with the possibility of using more than one AGV in the loops. Huang [45] proposed a new design concept of tandem AGV based on using of a transportation-center to link the transfer points in loops.

To test the viability of tandem configurations, Farling et al. [46] performed a simulation study to evaluate the impact of system size, machine failure rate and unload/load time on the performance of three AGV configurations, namely traditional (parallel unidirectional flows), the tandem flow-path and the tandem loop. In a tandem loop flow-path, there exists an express loop which connects all the loops (see [47]). Ross et al. [47] and Choi et al. [48] conducted some experiments to compare a tandem AGVS with a conventional AGVS. They compared the performance measures of the two systems in various conditions such as production, utilizations of vehicles, and mean production times, etc. Ho and Hsieh [49] proposed a design methodology for tandem AGV systems with multiple-load vehicles. Their goal was to devise a design methodology that could achieve the workload-balance between vehicles of different loops, minimize the inter-loop flow, and minimize the flow distance. They sequentially solved these problems in order to attain a complete design for the tandem AGV system. Simulation and Simulated Annealing were applied to solve the problems.

The application of metaheuristic algorithms in designing tandem paths is limited to only two papers. Most of the previous studies have proposed heuristic algorithms for the design of tandem routes. Only in [40] and [49], tabu search and simulated annealing have been applied as design procedures. In [40] hierarchical clustering was applied to partition departments in loops and tabu search was only used in order to find a single loop path for each zone. Thus, the application of metaheuristics as design procedures in tandem systems has not yet been fully considered. Table 1 characterizes problem specifications and solution techniques of the studies in the area of tandem path design. We selected the original definitions of tandem AGVS proposed in [17], as our problem definition and developed our algorithms based on them. Considering this, their proposed heuristic was the only existing algorithm which can be used as a base algorithm in comparing the performance of the newly developed algorithms.

The aim of this paper is to develop a TS and a GA procedure for designing routes in tandem configuration. Given a grid layout, a route sheet and the number of loops as inputs, the problem is to assign workstations to loops without allowing any overlap between loops, and in a way that the maximum workload of the system is minimized. As mentioned above, the proposed algorithms will be compared to the partitioning heuristic of [17], referred to as the base algorithm, using randomly generated test problems. This paper is organized as follows: a brief description of the problem and its assumptions are presented in Sect. 1; the TS and GA algorithms and their characteristics are discussed in Sect. 3, and Sect. 4 reports computational results; finally, conclusions are presented in Sect. 5.

2 Problem definition

Tandem AGV systems were first introduced by Bozer and Srinivasan in [15, 16]. Although the tandem configuration can be used both in warehousing and manufacturing, it was mainly defined and developed for the latter case. The system proposed by Bozer and Srinivasan in [17] is defined on a grid layout (Fig. 1). Each workstation is presented as a single point and may represent a machine, or a group of machines, such as a cell or a department.

The problem of configuring a tandem AGV system consists of partitioning a set of N workstations into several independent single AGV loops (zones). Additional P/D stations called transfer points are introduced to provide an interface between adjacent loops. Transfer points are connected to each other by/using conveyors. Figure 2 illustrates a typical tandem configuration.

The aim is to partition the workstations of a layout in such a way that each station is assigned to only one loop,

Table 1 Tandem flow path design procedures

Reference	Path direction	Layout	Number of AGVs in loops	AGV Type	Number of loops	Objective function	Solution technique
Bozer and Srinivasan (1992)	Bidirectional	Fixed	Single	Unit load	Fixed	Minimax of system workload	Set-partitioning algorithm
Hsieh and Sha (1996)	Unidirectional	Variable	Single	Unit load	Variable	Minimax of flow Between loops	Heuristic algorithm
Aarab et al. (2001)	Unidirectional	Fixed	Single	Unit load	Variable	Min of flow between and within loops	Hierarchical clustering, tabu search
Yu ad Egbelu (2001)	Bidirectional	Fixed	Single	Unit load	Fixed	Min of zones balance zone workload	Heuristic partitioning algorithm
Ho and Hsieh (2004)	Unidirectional	Variable	Single	Multiple load	Variable	Workload balance min of loop flow min of flow distance	Simulation, simulated annealing
Kim et al. (2003)	Unidirectional	Fixed	Single	Multiple load	Fixed	Minimax of system workload	Markov chain, set set-partitioning algorithm

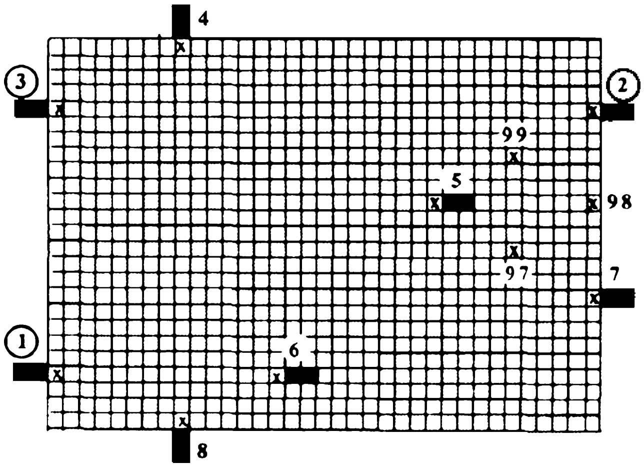

Fig. 1 A typical Grid layout ([17])

the workload of the AGVs associated with the material flow within and between the loops does not exceed the AGV capacity, and the workload is evenly distributed among all loops. The workload factor of each AGV, denoted by ω, is the proportion of time a vehicle is busy, either loaded or empty. This is the building block of the system and must be calculated for each loop.

The assumptions made by Bozer and Srinivasan in [17] are as follows. There are two types of workstations: input/output stations and process stations, where actual processing takes place. Transfer points are also considered as I/O stations. Every station has an I/O queue. A bidirectional single load AGV is used in each loop. When loaded, the AGV follows the shortest path to the destination station, and when empty it uses the FEFS (first encountered first served) empty dispatching rule, which will never leave the AGV idle. When tandem AGVSs were proposed for the first time, the FEFS was the only dispatching rule suggested [50].

Additional limitations are as follows: intersections and overlaps between loops are forbidden, and the number of

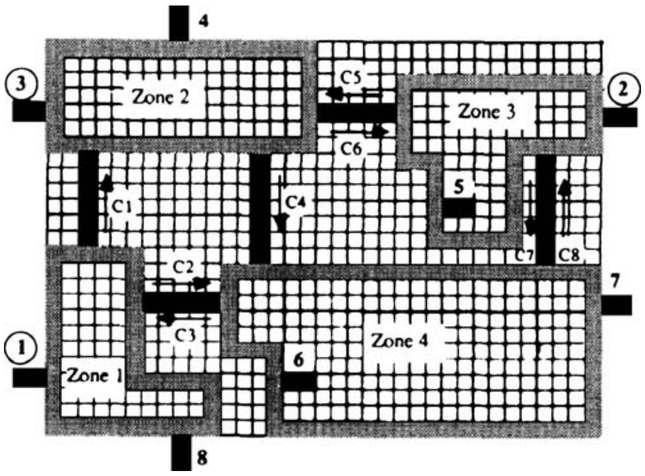

Fig. 2 A typical tandem configuration ([17])

loops must be at least two. The latter can be determined as an input, or can be obtained through the designing process. Here, the number of loops is given at the beginning of the algorithm.

3 Development of tabu search and genetic algorithm

A tabu search is a single solution metaheuristic search algorithm, which was first introduced by F. Glover (see [51–53]). It is a local search mechanism, which employs a short-term memory called a tabu list, to prevent cycling in local solutions. GA is a population based metaheuristic which employs the natural selection and natural genetics mechanisms to develop solutions to problems (see [54]). According to [53], TS, SA and GA was singled out by the Committee on the Next Decade of Operations Research as "extremely promising" for the future treatment of the practical applications. Moreover, GA and specially TS have shown good suitability for vehicle routing problems. Thus, TS and GA have been selected to solve the problem concerned in this paper. In this section, the developed algorithms based on TS and GA are described.

3.1 Feasibility conditions

A solution is called feasible if (1) no overlap or intersection exists among loops, (2) each station is assigned to only one loop, and (3) the workloads are less than 1 (or a reasonable value less than 1).

The algorithm proposed by Bozer and Srinivasan [17] does not have a specific mechanism to check the overlaps between the loops. The TS algorithm checks the intersection of loops for (before making any move) every move and selects only non-overlapping loops. The GA checks the intersection of loops created by genetic operators, using the penalty object. It tries to mate only those stations in each loop having zero penalty objects. Therefore, after a couple of iterations any intersection will be eliminated. For the sake of simplicity, the initial routes of loops are considered as the Euclidean traveling salesman route of the stations in the loop. Figure 3 shows a typical infeasibility in the presence of overlapping loops.

3.2 Number of loops

In the heuristic algorithm of Bozer and Srinivasan in [17], the number of loops is given as an input. This also applies to the TS and GA algorithms.

3.3 Developed tabu search algorithm

In this section the developed tabu search algorithm is described.

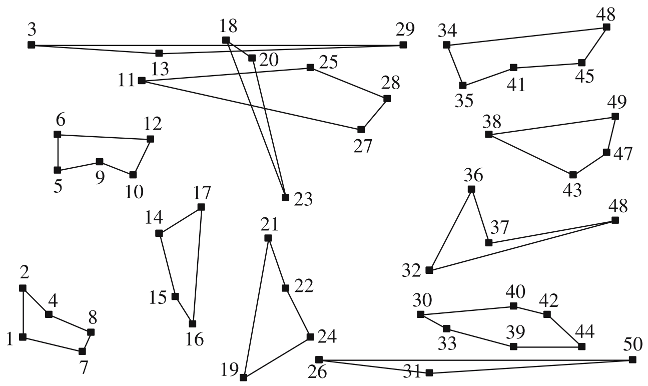

Fig. 3 A typical infeasible solution

3.3.1 Neighborhood structure

The neighborhood of a solution is simply obtained by removing a station from one loop and adding it to another, provided that it does not create any intersection and the workload of loops does not exceed 1. A current solution corresponding to a partition of the set of workstations in L loops can be represented as follows:

$$S = \{P_1, \ldots, P_i, \ldots, P_L\};\ i = 1, \ldots, L \tag{1}$$

Consider station s from the set of stations in loop P_i, where $st \in P_i$. Also, consider solution S' in the neighborhood of solution S:

$$S' = \{P'_1, \ldots, P'_i, \ldots, P'_L\};\ i = 1, \ldots, L \tag{2}$$

Solution S' is obtained by moving station s from loop P_i to loop P_j in solution S. In other words

$$P'_i = P_i - \{s\} \tag{3}$$

$$P'_j = P_j \cup \{s\}; j \neq i, \tag{4}$$

The feasible move m_{ij} is characterized by the transmission of station s from loop i to loop j, subject to the workload constraint. The neighborhood of S is the set of all feasible solutions that can be reached from S by applying moves m_{ij}.

A typical neighborhood structure is shown in Fig. 4. Figure 4a shows the possible moves for station 11, and Fig. 4b shows the new solutions resulting from possible moves of station 11 to 3 loops.

3.3.2 Objective function and evaluation of neighborhood

The evaluation method of neighborhood structure is based on the objective function of the integer linear programming model presented by Bozer and Srinivasan in [17]. The objective is to minimize the maximum workload of all loops.

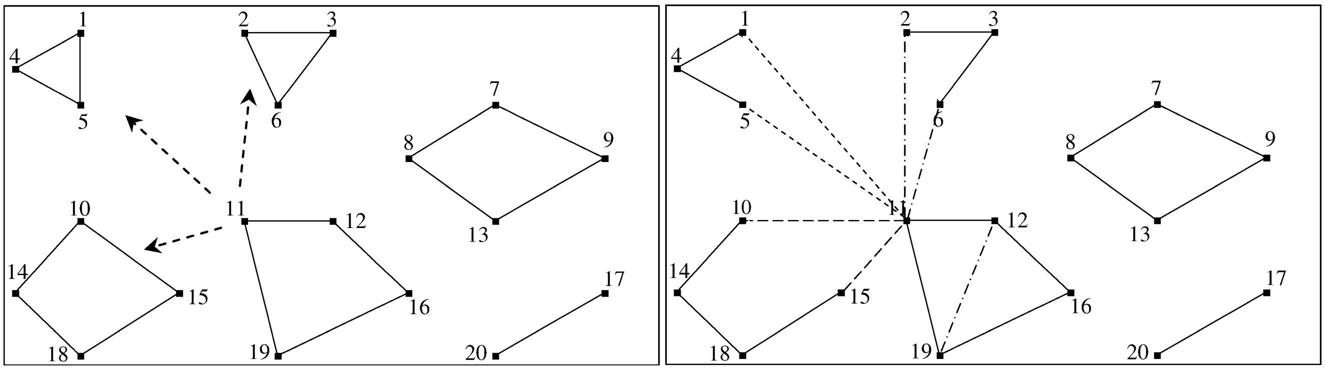

Fig. 4 A typical neighborhood structure

Thus, the basis of the search procedure is to select a loop with maximum workload and try to reduce its workload. The workload reduction is obtained by moving one of the stations of the selected loop to another loop, which causes leads to an increase in the workload of the second loop. The best move is the one resulting in the largest decrease in the workload of the selected loop and the smallest increase in the workload of the second loop. Thus, a simple expression is used as the evaluation criterion: workload reduction of the selected loop, minus workload increase of the second loop.

3.3.3 Generation of feasible solutions

The procedure of generating initial solutions consists of three phases:

a. Clustering the stations into L groups by means of k-means clustering method.
b. If necessary, removing workload infeasibilities.
c. Reducing the number of singleton stations.

In the first stage, a k-means clustering method is applied (see [55]). Applying the k-means clustering method for generating the initial partition ensures that no intersections will occur between the created loops, since distance is used as the closeness measure. On the other hand, the work-stations in a loop should be reasonably close to each other, so that unnecessary vehicle trips are avoided.

The initial cluster centers are chosen randomly and the rectilinear distance is used as the closeness measure, due to the rectilinear shape of the final routes of the loops. The resulting clusters are then checked for workload feasibility. In case of infeasibility, a simple search method is used to reduce the workload of infeasible loops by moving some stations, following the defined neighborhood evaluation method and based on the objective function. The search method will stop as soon as all infeasible workloads are reduced to less than 1. In the last phase, the configuration is checked for the presence of singleton loops and if more than one is present, another algorithm is employed to reduce the number of singletons by adding stations from the adjacent loops.

3.3.4 Definition and selection of moves

According to the neighborhood definition, what meant by a move is moving a station from one loop to another one. Based on minimax objective function, the evaluation of possible moves in each of the iterations is restricted to the loop with maximum workload.

Not all loops must be considered when looking for possible moves since in most cases applying the move will lead to overlaps. As a rule of thumb, usually just a few adjacent loops are worth checking. In order to prevent unnecessary calculations and to improve efficiency, we only consider the closest adjacent six or seven loops, according to the distance of their geometrical centers to the geometrical center of the selected loop. It is not necessary to check all stations in the selected loop, because this will result in computational inefficiencies. Similarly, it seems that the best candidate stations to be moved are often those whose removal will result in the largest workload decrease. It was, therefore, decided to check only the four stations with the largest workload decreases as candidates for removal from the loop. The possible moves can be defined as follows.

Assume that P_O is the selected loop such that is has the maximum workload among existing loops. Also, let A_O be the subset of selected stations in loop P_O. The loops are selected as candidate destination loops for stations in A_O, denoted as P_{D_j} where j=1, ..., L_D , $j \neq i$ and L_D is the number of candidate loops. Based on the neighborhood definition, the inclusion of a station in another loop is allowed only if this does not lead to an infeasible solution. A destination loop workload threshold $\overline{\varpi}$ has been set to select the moves leading to the best possible solutions, by limiting the increase of the workload of the other loop. When evaluating possible moves, the moves that satisfy the threshold conditions are first considered. In the absence of such moves, this restriction is not taken into account and all possible moves are evaluated.

3.3.5 *Tabu restrictions and aspiration criterion*

The algorithm uses a fixed size tabu list. As a result of move definition, once a station is moved from one loop to another, it cannot be added to that loop again for the next θ iterations. However, when a potential tabu move leads to a solution better than the best solution found so far, its tabu status is revoked.

3.3.6 *Diversification*

Diversification is a commonly used strategy in TS. It is used to allow the search mechanism to escape from probable local minima. In our implementation, when no feasible move exists, the tabu list is emptied and the search restarts from the best known solution.

3.3.7 *Termination criterion*

The termination criterion is set as the maximum number of iterations performed since the best solution has changed. In our case this value was set to 20.

3.3.8 *Allowing infeasible solutions*

In some cases, all possible moves lead to an increased workload. In such situations, the search is allowed to proceed outside the feasible space, in the hope of reaching feasibility again at a later stage.

3.3.9 *Controlling singleton stations*

A control is imposed in order to limit the number of singleton stations in the configuration. This is done in two stages: in the generation of the initial solution and in the main body of the algorithm. After generating an initial feasible solution, the number of singleton stations is checked and if there is more than one, another subroutine is applied to reduce them.

Another control is imposed as a condition for accepting of the current solution as the best solution. The number of singleton stations in the current solution is checked and if it is more than one, the same algorithm is applied to reduce it. If the number of stations in the best solution is more than one, and the current solution has fewer singleton stations, it is accepted as the best solution, even if it has a worse objective function. In other words, a solution with fewer singleton stations is preferred when the number of singleton stations is more than one. Otherwise, if this number is less than or equal to one in the best solution found so far, the current solution is accepted as the best solution, when it has less than or equal to one singleton station and a better objective function, too.

3.3.10 *The detailed description of the Tabu Search algorithm*

The detailed description of the TS algorithm is as follows:

```
Phase 1   1) Generate an initial solution $S_0$, and set $S: = S^*$, $f(S): = f(S^*)$;
          2) Set $\tau: = 0$ ($\tau$ is the number of the last iteration with
             an improvement in the objective function);
          3) Set iteration counter to zero: $t: = 0$.
Phase 2   while $t- \tau<20$, repeat
          1) $t := t + 1$;
          2) Choose the loop with maximum workload ($P_O$);
          3) Choose the best move and update $f(S)$, $S$ (if any
             moves exist considering the workload threshold,
             select the best move);
          4) If no possible moves exists, then set $S:=S^*$, $f(S): =
             f(S^*)$; and $TL = \varnothing$ (diversification);
          5) Update the tabu list;
          6) If $\sigma(S) > 1$ ($\sigma(S)$ is the number of singleton
             stations in solution $S$), attempt to reduce it;
          7) If $\sigma(S) > 1$ but $\sigma(S) < \sigma(S^*)$, then set $S: = S^*$, $f(S): =
             f(S^*)$; else if $\sigma(S^*) = 1$, $\sigma(S) = 1$ and $f(S) < f(S^*)$,
             then set $S: = S^*$, $f(S): = f(S^*)$.
```

3.4 Developed genetic algorithm

In this section the developed genetic algorithm is described.

3.4.1 *Solution presentation*

Each solution is presented by a chromosome that defines the assignment of stations to loops. A chromosome is an N-element vector, where N is the number of stations. Assuming that the index of each element represents all of the N stations, the values of elements (genes) define the loop number to which the station is assigned. Figure 5 shows a typical presentation of a solution for an eight-station problem with four loops.

3.4.2 *Fitness function*

As in tabu search algorithm, the objective function is to minimize the maximum workload of the system. Thus, the

Station number	1	2	3	4	5	6	7	8
Chromosome	1	1	2	3	5	2	3	4

Fig. 5 A typical presentation of chromosome

fitness function of a chromosome is the maximum workload of the loops, which can be obtained by calculating the workload of each loop. The algorithm tries to select chromosomes with the minimum workload from the maximum workloads just mentioned.

3.4.3 Initial population generation

A heuristic algorithm has been developed to generate the required population. Almost, the whole answer space of the problem will be generated by the heuristic algorithm developed for this purpose. After creating possible solutions, duplications and infeasible solutions are removed and the remaining space is used as the initial population. Then, in the same number of chromosomes will be selected randomly from the mentioned space.

3.4.4 Selection

In this model, the probability of selecting better choices is more than the worse ones, if using roulette wheel. A fitness function is assigned to each loop. The chromosome with higher fitness value (i.e., higher maximum workload), will receive a lower weight to be selected. Based on these weights, a couple of chromosomes will be chosen randomly for the next generation. Chromosomes with lower weights have less opportunity to be selected, but the associated probability is not zero. There will couple of chromosomes at the same size of first population, at the end. Obviously, some of these have been already chosen, while others have not.

3.4.5 Genetic operators

In this algorithm, the one-point crossover operator is used. According to the mentioned probabilistic selection method, a number of chromosomes will be chosen from the first population for the next generation, so that they can be mated to create children (offspring). The genetic algorithm tries to create new children by switching a couple of genes in the selected chromosomes, so that it can improve the workloads. Figure 6 shows a typical crossover using the one-point operator. The dashed line shows the crossover point. After generating all the children, they will be compared with their own parents. The better ones will be elected for the new generation. It will take a short time to get an answer, because the locations of changed genes are randomly selected.

After all, to avoid local answers, the algorithm uses the mutation operator. Based on the first population size, number of stations and the probability of mutation (P_{mut}), specified at the beginning, a known number of genes will be nominated for mutation.

Station number	1	2	3	4	5	6	7	8
Parent 1	2	1	3	1	4	3	4	3
Parent 2	4	3	1	3	1	4	2	2
After corss-over								
Child 1	2	1	3	3	1	4	2	2
Child 2	4	3	1	1	4	3	4	3

Fig. 6 A typical presentation of crossover

3.4.6 Termination criteria

Since no mathematical model exists for tandem configuration design , the optimum solutions are not known. Also, due to the short calculation time of the "maximum number of generations" termination criteria, is used. In our case this value was set to 50 and in most cases the algorithm gets to the best solution.

To avoid similar iterations, the algorithm will be terminated if the best solution does not improve after a specified number of generations. In this case, this number is set equal to 3.

3.4.7 The detailed description of the genetic algorithm

Based on the above definitions, the developed algorithm is as follows:

Phase 1
1) Generate the initial answer space using the heuristic algorithm.
2) Randomly select a couple of chromosomes with the same size as the size of the first population from this space.
3) Set iteration counter to zero: $t: = 0$.

Phase 2: while $t <$ "maximum number of generations", repeat

1) $t := t + 1$;
2) If $t >= 15$, check each of the 3 sequential generations for any improvement. If there is no better solution, then stop.
3) Upon the calculated fitness function, assign a weight to each chromosome. (This weight has a reverse relation with the fitness function. A chromosome with higher fitness function will have a lower weight.)
4) Multiply the weight of any chromosome in the selection group with intersection in its loops, by a penalty probability (equal to 0.1), so that the total weight will be in its lowest range.
5) Calculate the cumulative summation of weights for each chromosome.
6) Produce a couple of random numbers between 0 and 1, with the same size as the first population. Compare them with the cumulative summation in step 5. Any span of the two cumulative summation

numbers with a randomly number, will be selected. (Each span presents a chromosome.)

7) Create a population for generating children and the next generation, with the same size as the first population.

Phase 3 1) Produce a couple of random numbers between 0 and 1, with the same size as the first population. Compare them with the cross probability (Pc=0.3) and save the index of any of number less than Pc.

2) Separate chromosomes with the same indices from the selected population of step 1. Mate them two by two.

3) Randomly specify the location of each cross. Brake chromosomes and mix them with each other.

Phase 4 1) Compute the workload and the fitness function of the new chromosomes.

2) Put the new chromosomes and their parents together, and sort them by their fitness function value increasingly.

3) Choose the best group of chromosome with the same size of the original to generate the new generation.

Phase 5 1) Specify the number of genes to be mutated (NP), based on the number of genes and Pmut ($Pmut$=0.01).

2) Randomly select NP genes for mutation. Specify their location in the chromosomes.

3) Add one to the number of genes, presenting the loop that station belongs to.

Phase 6 1) Specify the workload of each loop in the new chromosomes and specify their fitness functions

2) Check the feasibility of the chromosomes. For any chromosome with an intersection, assign a penalty probability (0.1). Consider these penalties in computing weights.

4 Computational results

Since no benchmark problem instances exist for the tandem configuration problem, some randomly generated problems were used to test the algorithms.

4.1 Test problems

The test problems were generated for five types of grid layouts with 10, 20, 30, 40 and 50 stations. For each layout size, three types of from-to flow charts were randomly built with 0.2, 0.25 and 0.5 densities. Flow values (units/hour) were chosen randomly between 0.05 and 0.3. The specifications for the AGV were obtained from [16]. The speed of the AGV (empty or loaded) and the time required to pick up or deliver a load were set to 15 grid units/minute and 0.2 minutes respectively. For each of the 15 problems, four random instances were generated. In total, 60 problems were solved.

4.2 Solving the problems

Each of the test problems was solved for three levels L (number of loops). The L values were initially derived from the base algorithm and then were applied to the solution procedure of the algorithms. These L values are:

- $L_{\max}$: the maximum possible value for L. Assuming that each station can form a feasible loop with at least one of its adjacent stations, this value will be equal to $[0.5N]$;
- $L_{\min}$: the minimum possible value for L, at which the objective function doest not exceed 0.7 (the selected threshold for workload in the base algorithm);
- L_{average}: the mean of $L_{\max}$ and $L_{\min}$.

Based on the suggestions provided in [17], in order to reduce the run time of the integer linear programming model, an estimated threshold z_H was used to eliminate unnecessary loops. This threshold was set to 0.7 for $L_{\min}$, was obtained from the average and maximum workload for loops with two or three stations for $L_{\max}$, and finally for L_{average}, it was set to a value between the z_H of $L_{\max}$ and $L_{\min}$, or the average workload for loops with sizes equal to $[N/L]$. Since at most one singleton station in the final configuration, obtained from the TS algorithm, was assumed, the same assumption was made for the base algorithm as well.

The first two phases of the base algorithm, generating subsets of stations and eliminating some of them, were coded by Matlab 6.5 and the IP model was coded using LINGO 8.0 software [56]. The TS and GA algorithms were coded using Matlab 6.5. It was found that in most cases the effective size of the tabu list is $[N/5]\pm2$. For the TS algorithm, three randomly selected initial solutions were used for each problem, except for some small problems, where the number did not reach three. For the GA algorithm, each problem was solved four times for each loop number.

4.3 Comparison of algorithms

Tests were carried out on a 2.00 GHz Intel Pentium 4, with 256 MB RAM. The summaries of the test results for TS and GA are shown in Tables 2 and 3. The statistics of Tables 2 and 3 are summarized in Tables 4 and 5. For small

Table 2 Summary of the test results for TS

N	d	L	Infeasible solutions of the base Alg	Better solutions		Worse solutions		Equal solutions		Average deviation of the TS objective function from the base algorithm		Best improvement in the objective function	Average run time (seconds)	
Number of stations	Density of FT chart	Number of loops		Number	Percent	Number	Percent	Number	Percent	Positive deviation	Negative deviation		Base Alg.	TS
10	0.2	Max	0	0	0%	0	0%	11	100%	–	–	–	212.9	15.7
		Average	0	4	44%	0	0%	5	56%	–	−6.1%	−9.4%	233.7	24.5
		Min	1	4	67%	0	0%	2	33%	–	−0.4%	−0.5%	214.2	57
	0.25	Max	0	2	20%	1	10%	7	70%	0.7%	−14.7%	−15.4%	281.1	20.1
		Average	1	0	0%	0	10%	7	100%	–	–	–	298.8	23
		Min	0	0	0%	0	0%	6	100%	–	–	–	282.3	65.9
20	0.5	Max	0	0	0%	0	0%	12	100%	–	–	–	75.7	17.4
		Average	0	0	0%	2	20%	8	80%	6.8%	–	–	81	25.1
		Min	0	2	29%	0	0%	5	71%	–	−2.8%	−2.8%	73	66.1
	0.2	Max	2	2	17%	1	8%	9	75%	3.1%	−0.6%	−6.0%	160.1	52.8
		Average	0	11	92%	1	8%	0	0%	2.2%	−6.7%	−13.9%	162.3	144.8
		Min	0	4	67%	0	0%	2	33%	–	−6.3%	−10.8%	171.2	363
	0.25	Max	1	3	25%	1	8%	8	67%	2.2%	−1.0%	−1.0%	109.7	60.2
		Average	0	2	17%	1	8%	9	75%	3.8%	−1.0%	−1.0%	109.1	107.1
		Min	1	5	45%	1	9%	5	45%	6.7%	−5.1%	−6.5%	116.9	239.1
	0.5	Max	1	6	50%	1	8%	5	42%	0.2%	−11.4%	−17.1%	36.4	61
		Average	2	6	50%	0	0%	6	50%	–	−6.6%	−11.0%	37.3	91.6
		Min	1	8	67%	3	25%	1	8%	5.7%	−8.9%	−17.4%	40.5	198.7
30	0.2	Max	2	9	75%	0	0%	2	25%	–	−4.5%	−14.0%	166	92.7
		Average	3	10	83%	1	8%	1	8%	3.1%	−4.9%	−12.5%	145.3	137.8
		Min	2	11	92%	0	0%	1	8%	–	−8.0%	−19.0%	165.2	309.7
	0.25	Max	1	3	25%	1	8%	8	67%	2.0%	−4.8%	−5.0%	150.5	78.7
		Average	1	8	67%	3	25%	1	8%	4.5%	−5.3%	−12.8%	154.1	135.4
		Min	0	12	100%	0	0%	0	0%	–	−2.7%	−11.7%	160.7	427.1
	0.5	Max	1	7	58%	4	33%	1	8%	6.3%	−7.9%	−19.1%	50.4	87.6
		Average	1	8	58%	3	25%	2	17%	2.3%	−7.5%	−11.1%	44.9	151.6
		Min	2	10	83%	2	17%	0	0%	1.8%	−5.7%	−11.8%	50.1	244.6
40	0.2	Max	3	5	42%	5	42%	2	17%	2.9%	−9.4%	−19.1%	186.4	98.8
		Average	3	9	75%	3	25%	0	0%	4.8%	−4.3%	−15.9%	196.3	184.3
		Min	0	12	100%	0	0%	0	0%	–	−11.4%	−22.5%	202.9	547.6
	0.25	Max	2	9	75%	3	25%	0	0%	5.4%	−4.1%	−8.8%	131.3	96
		Average	4	6	50%	2	17%	4	33%	7.0%	−5.1%	−9.6%	137.2	210.7
		Min	0	11	92%	1	8%	0	0%	4.6%	−5.6%	−12.1%	155.4	358.1
	0.5	Max	1	7	58%	5	42%	0	0%	6.8%	−5.2%	−14.0%	48.2	114.9
		Average	2	7	58%	5	42%	0	0%	2.3%	−3.3%	−4.6%	50.3	159.1

Table 2 (continued)

N	d	L	Infeasible solutions of the base Alg	Better solutions		Worse solutions		Equal solutions		Average deviation of the TS objective function from the base algorithm		Best improvement in the objective function	Average run time (seconds)	
Number of stations	Density of FT chart	Number of loops		Number	Percent	Number	Percent	Number	Percent	Positive deviation	Negative deviation		Base Alg.	TS
		Min	1	10	83%	2	12%	0	0%	6.1%	−5.5%	−8.8%	53.7	252.2
50	0.2	Max	3	7	58%	5	42%	0	0%	2.4%	−9.5%	−14.8%	173.6	143
		Average	4	11	92%	1	8%	0	0%	3.3%	−10.0%	−21.7%	191.2	253.9
		Min	0	11	92%	1	8%	0	0%	1.5%	−7.1%	−14.7%	219.7	477
	0.25	Max	3	12	100%	0	0%	0	0%	–	−8.4%	−21.4%	128.6	161.1
		Average	4	11	92%	1	8%	0	0%	0.9%	−9.4%	−16.0%	151.2	256.5
		Min	1	12	100%	0	0%	0	0%	–	−11.5%	−21.4%	160.5	405.8
	0.5	Max	4	10	83%	2	12%	0	0%	7.8%	−6.8%	−16.9%	63.3	170
		Average	3	6	50%	4	33%	2	17%	1.2%	−3.8%	−5.9%	56	219.3
		Min	2	10	83%	2	12%	0	0%	0.7%	−7.80%	−12.6%	66.9	221.3

Table 3 Summary of the test results for GA

N	d	L	Infeasible solutions of the base Alg	Better solutions		Worse solutions		Equal solutions		Average deviation of the TS objective function from the base algorithm		Best improvement in the objective function	Average run time (seconds)	
Number of stations	Density of FT chart	Number of loops		Number	Percent	Number	Percent	Number	Percent	Positive deviation	Negative deviation		Base Alg.	TS
10	0.2	Max	0	1	6%	3	19%	12	75%	−4.24	6.10	−12.68	212.9	79.9
		Average	0	6	38%	1	6%	9	56%	−2.61	0.53	−9.42	233.7	126.5
		Min	1	6	38%	0	0%	10	63%	−0.29	0.00	−0.53	214.2	79.37
	0.25	Max	0	4	25%	2	13%	10	63%	−4.87	3.53	−15.57	281.1	65.65
		Average	1	0	0%	1	6%	15	94%	−0.05	9.44	−0.06	298.8	135.8
		Min	0	1	6%	0	0%	15	94%	−0.03	0.00	−0.08	282.3	94.96
	0.5	Max	0	0	0%	0	0%	16	100%	−0.03	0.00	−0.04	75.7	46.37
		Average	0	0	0%	1	6%	15	94%	0.00	1.04	0.00	81	67.37
		Min	0	3	19%	0	0%	13	81%	−2.77	0.00	−2.77	73	62.85
20	0.2	Max	2	3	19%	3	19%	10	63%	−0.57	14.25	−0.57	160.1	109.3
		Average	0	15	94%	0	0%	1	6%	−6.85	0.00	−13.90	162.3	133.2

		Min	0	12	75%	0	0%	4	25%	−5.26	0.00	−10.76	171.2	130.9
	0.25	Max	1	2	13%	5	31%	9	56%	−0.35	48.23	−0.96	109.7	109.5
		Average	0	3	19%	3	19%	10	63%	−1.05	4.53	−1.05	109.1	94.63
		Min	1	7	44%	6	38%	3	19%	−5.58	8.65	−6.47	116.9	117.9
	0.5	Max	1	6	38%	2	13%	8	50%	−5.84	13.69	−17.07	36.4	50.53
		Average	2	4	25%	1	6%	11	69%	−3.49	14.30	−10.96	37.3	36.58
		Min	1	11	69%	2	13%	3	19%	−5.13	7.12	−17.44	40.5	35.51
30	0.2	Max	2	8	50%	0	0%	8	50%	−3.32	2.93	−14.01	166	118.4
		Average	3	13	81%	2	13%	1	6%	−2.93	0.29	−7.11	145.3	136.5
		Min	2	12	75%	0	0%	4	25%	−6.01	0.00	−10.08	165.2	145.1
	0.25	Max	1	9	56%	5	31%	2	13%	−4.74	3.38	−4.74	150.5	203.5
		Average	1	9	56%	0	0%	7	44%	−2.91	0.00	−5.72	154.1	180.5
		Min	0	11	69%	1	6%	4	25%	−1.68	0.00	−5.09	160.7	190.9
	0.5	Max	1	7	44%	5	31%	4	25%	−7.64	5.82	−19.11	50.4	105.2
		Average	1	6	38%	1	6%	9	56%	−8.55	1.22	−11.11	44.9	107.8
		Min	2	11	69%	2	13%	3	19%	−3.39	1.83	−8.95	50.1	96.74
40	0.2	Max	3	6	38%	6	38%	4	25%	−8.24	0.55	−19.09	186.4	188.7
		Average	3	8	50%	4	25%	4	25%	−2.90	4.25	−4.74	196.3	184.6
		Min	0	11	69%	0	0%	5	31%	−9.08	0.00	−20.26	202.9	197.3
	0.25	Max	2	9	56%	4	25%	3	19%	−1.05	5.29	−3.00	131.3	190.7
		Average	4	4	25%	4	25%	8	50%	−3.62	0.73	−6.25	137.2	190.6
		Min	0	10	63%	0	0%	6	38%	−3.99	0.00	−5.14	155.4	191.9
	0.5	Max	1	9	56%	3	19%	4	25%	−4.50	5.00	−12.16	48.2	189.7
		Average	2	7	44%	6	38%	3	19%	−2.46	1.92	−3.21	50.3	216.9
		Min	1	9	56%	3	19%	4	25%	−4.03	4.07	−7.97	53.7	196.3
50	0.2	Max	3	10	63%	4	25%	2	13%	−7.69	1.96	−10.42	173.6	252
		Average	4	15	94%	0	0%	1	6%	−8.35	0.00	−19.32	191.2	283.7
		Min	0	10	63%	0	0%	6	38%	−3.23	0.00	−6.49	219.7	278.3
	0.25	Max	3	15	94%	0	0%	1	6%	−5.74	0.00	−9.67	128.6	269.5
		Average	4	10	63%	2	13%	4	25%	−8.31	0.85	−9.99	151.2	262.9
		Min	1	11	69%	0	0%	5	31%	−9.57	0.00	−19.15	160.5	276.2
	0.5	Max	4	8	50%	2	13%	6	38%	−4.15	6.37	−11.24	63.3	251.4
		Average	3	5	31%	4	25%	7	44%	−2.58	1.03	−5.12	56	232.5
		Min	2	10	63%	2	13%	4	25%	−5.91	0.68	−11.41	66.9	219.3

Table 4 Summary of solution cases for TS

Type of case	Number of instances	Percent of instances	Average deviation from the base algorithm	Maximum deviation from the base algorithm
Better objective function	301	60%	−6.3%	−22.5%
Worse objective function	68	13%	3.6%	13.3%
Equal objective function	134	27%	–	–
Total	503	100%	–	–

Table 5 Summary of solution cases for GA

Type of case	Number of instances	Percent of instances	Average deviation from the base algorithm	Maximum deviation from the base algorithm
Better objective function	337	47%	−8.31%	−20.26%
Worse objective function	90	12%	3.99%	59.13%
Equal objective function	293	41%	–	–
Total	720	100%	–	–

Table 6 Summary of t-tests for objective functions

	TS against base algorithm	GA against base algorithm	TS against GA
H_0	$\grave{i}_{TS} - \grave{i}_{Base}=0$	$\grave{i}_{GA}-\grave{i}_{Base}=0$	$\grave{i}_{TS}-\grave{i}_{GA}=0$
H_1	$\grave{i}_{TS} - \grave{i}_{Base}>0$	$\grave{i}_{GA}-\grave{i}_{Base}>0$	$\grave{i}_{TS}-\grave{i}_{GA}>0$
Critical Area	$t = t_{critical}=1.65$	$t = t_{critical}=1.65$	$t = t_{critical}=1.65$
t-value	−0.6935	−0.3247	−0.3533
Test result	accept	Accept	accept

problems, the objective functions of the two algorithms are often equal to the base algorithm, which might be the value of optimal solution. The percentage of better solutions significantly increases as a function of the instance size, *N*.

To test the performance of the algorithms against each other, some statistical tests were carried out. Since the variances were unknown, first F-tests were carried out to check the equality of variances. All tests showed that the equality of variances could not be accepted. Based on this result, t-test was run to compare means with the assumption of unequal variances of samples at an α-level of 0.05. The hypotheses and test results are shown in Table 6. Based on the test results, it was accepted that TS and GA are better than or the same as base algorithm and that TS is better than or the same as GA.

Table 7 Summary of t-tests for run times

	TS against base algorithm	GA against base algorithm	TS against GA
H_0	$\grave{i}_{TS}-\grave{i}_{Base}=0$	$\grave{i}_{GA}-\grave{i}_{Base}=0$	$\grave{i}_{TS}-\grave{i}_{GA}=0$
H_1	$\grave{i}_{TS}-\grave{i}_{Base}>0$	$\grave{i}_{GA} - \grave{i}_{Base}>0$	$\grave{i}_{TS}-\grave{i}_{GA}>0$
Critical area	$t=t_{critical}=1.65$	$t=t_{critical}=1.65$	$t=t_{critical}=1.65$
t-value	2.852	6.51	−2.005
Test result	Reject	Reject	Accept

The number of loops (*L*) has a significant effect on the running time of both TS and GA algorithms. The average run time of the algorithms grows as the number of loops increases. The explanation is that as the number of stations in the loops increases, more time is required to compute the workloads. In comparison with the base algorithm, the algorithms are often faster for lower flow densities and higher *L* values. In general, it seems that on larger problems the base algorithm tends to be faster, but the latter algorithms remain preferable because they always produce feasible solutions.

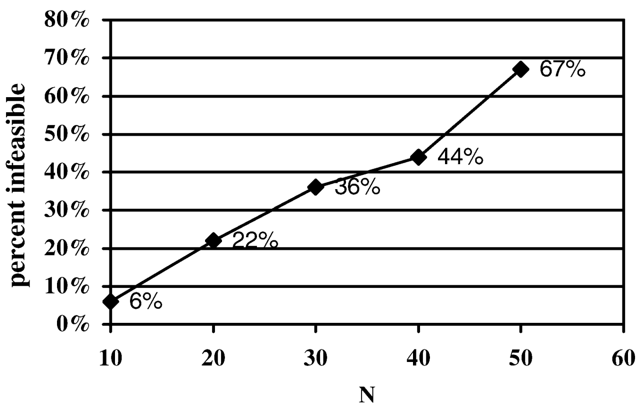

Fig. 7 Percentage of infeasible solutions by base algorithm

Statistical tests were conducted to compare the computational times of algorithms. Tests were similar to those of the objective functions. All tests showed that the equality of the variances could not be accepted. Based on this result, t-test was run to compare means with the assumption of unequal variances of samples at an α-level of 0.05. Hypotheses and test results are shown in Table 7. Based on the test results, it was rejected that TS and GA run times are better than or equal to the base algorithm's run times, but it was accepted that TS run times are better than or equal to the GA's run times.

The main advantage of the TS and GA algorithms compared to the base algorithm, in addition to improved solutions, is the feasibility of solutions. The TS and GA algorithms prevent or eliminate intersections between the loops, but the base algorithm has no specific mechanism to ensure the independence of loops in the final configuration. As a result, a number of solutions generated by this algorithm are infeasible. The percentage of infeasible solutions according to the problem size is shown in Fig. 7. The percentage of infeasible solutions increases with problem size. For the largest problem size solved, more than 60% of solutions are infeasible. It can be inferred that the base algorithm is not successful in solving problems with more than 20 stations. In any case, the infeasibilities are present even in smaller problems.

5 Conclusions

We have proposed new TS and GA algorithms for designing AGV routes in a tandem configuration. Our heuristics were compared to the base heuristic of Bozer and Srinivasan in [17]. The three algorithms were run on 60 randomly generated problems at three levels of loop numbers. Results show that our algorithms are capable of producing better solutions and the amount of improvements in the objective function with respect to the base algorithm tends to be higher as the problem size increases. Statistical tests showed that the hypotheses of better or equal objective functions for TS and GA in respect to base algorithm and for TS against GA cannot be rejected. Similar tests showed that the hypotheses of shorter run times for TS and GA in respect to base algorithm can not be accepted, but hypothesis of TS run times being better than GA is accepted. The main advantage of our TS algorithm is the impossibility of generating overlapping loops. Our GA algorithm applies a penalty object to eliminate intersections. Results show that as the problem size increases, the likelihood of generating infeasible solutions tends to be very high for the base algorithm.

Acknowledgments We would like to thank Farhang Fasihi and Nasim Akbarzadeh for their valuable comments on this work; their suggestions are appreciated. The authors would also like to thank two anonymous referees for their constructive suggestions, which improved the content substantially. The authors are responsible for all results and opinions expressed in this paper.

References

1. Tompkins JA, White JA, Bozer YA, Tanchoco JMA (2003) Facilities planning, 3rd edn. Wiley, Chichester
2. Asef-Vaziri A, Laporte L (2005) Loop based facility planning and material handling. Europ J Oper Res 164:1–11
3. Gaskin RJ, Tanchoco JMA (1987) Flow path design for automated guided vehicle system. Int J Prod Res 25:667–676
4. Gaskin RJ, Tanchoco RMA, Taghaboni F (1989) Virtual flow paths for free ranging automated guided vehicle systems. Int J Prod Res 27:91–100
5. Kaspi M, Tanchoco JMA (1990) Optimal flow path design of unidirectional AGV systems. Int J Prod Res 28:1023–1030
6. Venkataramanan MA, Wilson KA (1991) A branch- and- bound algorithm for flow path design of automated guided vehicle systems. Naval Res Logist Quart 38:431–445
7. Sinriech D, Tanchoco JMA (1991) Intersection graph method for AGV flow path design. Int J Prod Res 29:1725–1732
8. Kouvelis P, Gutierrez GJ, Chiang WC (1992) Heuristic unidirectional flow path design approach for automated guided vehicle systems. Int J Prod Res 30:1327–1351
9. Seo Y, Egbelu PJ (1995) Flexible guide path design for automated guided vehicle systems. Int J Prod Res 33:1135–1156
10. Sun XC, Tchernev N (1996) Impact of empty vehicle flow on optimal flow path design for unidirectional AGV systems. Int J Prod Res 34:2827–2852
11. Zanjirani Farahani R, Tari FG (2001) Optimal flow path designing of unidirectional AGV systems. Intl J Engin Sci 12:31–44
12. Zanjirani Farahani R, Tari FG (2002) A branch & bound method for finding flow-path designing of AGV systems. IIE Trans A Basics 15:81–90
13. Kaspi M, Kesselman U, Tanchoco JMA (2002) Optimal solution for the flow path design problem of a balanced unidirectional AGV system. Int J Prod Res 40:349–401
14. Ko KC, Egbelu PJ (2003) Unidirectional AGV guide path network design: a heuristic algorithm. Int J Prod Res 41:2325–2343
15. Bozer YA, Srinivasan MM (1989) Tandem configurations for AGV systems offer simplicity and flexibility. Industr Eng 21:23–27
16. Bozer YA, Srinivasan MM (1991) Tandem configuration for automated guided vehicle systems and the analysis of single vehicle loops. IIE Trans 23:72–82
17. Bozer YA, Srinivasan MM (1992) Tandem AGV systems: a partitioning algorithm and performance comparison with conventional AGV systems. Europ J Oper Res 63:173–191
18. Lin JT, Chang CCK, Liu WC (1994) A load routing problem in a tandem-configuration automated guided vehicle system. Int J Prod Res 32:411–427
19. Laporte G, Zanjirani Farahani R, Miandoabchi E (2006) Designing an efficient method for tandem AGV network design problem using tabu search. Appl Math Comput (in press)
20. Tanchoco JMA, Sinriech D (1992) OSL-Optimal Single Loop guide paths for AGVs. Int J Prod Res 30:665–681
21. Sinriech D, Tanchoco JMA (1993) Solution methods for the mathematical models of single loop AGV systems. Int J Prod Res 31:705–726
22. Banerjee P, Zhou Y (1995) Facilities layout design optimization with single loop material flow path configuration. Int J Prod Res 33:183–203

23. Laporte G, Asef-Vaziri A, Sriskandarajah C (1996) Some application of the generalized traveling salesman problem. J Oper Res Society 47:1461–1467
24. Asef-Vaziri A, Laporte G, Sriskandarajah C (2000) The block layout shortest loop design problem. IIE Trans 32:724–734
25. Asef-Vaziri A, Dessouky M, Sriskandarajah C (2001) A loop material flow system design for automated guided vehicles. Intl J Flex Manuf Syst 13:33–48
26. Asef-Vaziri A, Laporte L, Ortiz R (2006) Exact and heuristic procedures for the material handling circular flow path design problem. Europ J Oper Res DOI 10.1016/j.ejor.2005.08.023
27. Zanjirani Farahani R, Laporte G, Sharifyazdi M (2005) A practical exact algorithm for the shortest loop design problem in a block layout. Int J Prod Res 43:1879–1887
28. Zanjirani Farahani R, Pourakbar M, Miandoabchi E (2006b) Developing exact and tabu search algorithms for simultaneously determining AGV loop and P/D stations in single loop systems. Int J Prod Res (in press)
29. Zanjirani Farahani R, Karimi B, Tamadon S (2006a) Designing an efficient method for simultaneously determining the loop and the location of the P/D stations using genetic algorithm. Int J Prod Res (in press)
30. Kim CW, Tanchoco JMA (1991) Conflict-free shortest-time bi-directional AGV routing. Int J Prod Res 29:2377–2391
31. Chhajed D, Montreuil B, Lowe T (1992) Flow network design for manufacturing systems layout. Europ J Oper Res 57:145–161
32. Rajagopalan S, Heragu SS, Taylor GD (2004) A lagrangian relaxation approach to solving the integrated pick-up/drop-off point and AGV flow path design problem. Appl Math Model 28:735–750
33. Sinriech D, Tanchoco JMA (1994) SFT-Segmented Flow Topology. In: Tanchoco JMA (ed) Material flow system in manufacturing. Chapman and Hall, London, pp 200–235
34. Sinriech D, Tanchoco JMA (1995) An introduction to the segmented flow approach to discrete material flow systems. Int J Prod Res 33:3381–3410
35. Sinriech D, Tanchoco JMA (1997) Design procedures and implementation of the Segmented Flow Topology (SFT) for discrete material flow systems. IIE Trans 29:323–335
36. Barad M, Sinriech D (1998) A petri net model for the operational design and analysis of Segmented Flow Topology (SFT) AGV system. Int J Prod Res 36:1401–1426
37. Vis IFA (2006) Survey of research in the design and control of automated guided vehicle systems. Europ J Oper Res 170:677–709
38. Hsieh LF, Sha DY (1996) A design process for tandem automated guided vehicle systems: the concurrent design of machine layout and guided vehicle routes in tandem automated guided vehicle systems. Integ Manuf Syst 7:30–38
39. Liu FH, Chen JT (1997) Analytical framework for designing the divided automated guided vehicles system. Int J Industr Engin 4 (2):90–102
40. Aarab A, Chetto H, Radouance L (1999) Flow path design for AGV systems. Stud Inform Contr 8(2):97–106
41. Yu W, Egbelu P (2001) Design of a variable path tandem layout for automated guided vehicle systems. J Manuf Syst 20:305–319
42. Kim KS, Chung BD, Jae M (2003) A design for a tandem AGVS with multi-load AGVs. Int J Adv Manuf Technol 22:744–752
43. Bozer YA, Lee CG (2004) Using existing workstations as transfer stations in tandem AGV systems. J Manuf Syst 23:229–241
44. Ventura JA, Lee CU (2001) Tandem loop with multiple vehicles configuration for automated guided vehicle systems. J Manuf Syst 20:153–165
45. Huang C (1997) Design of material transportation system for tandem automated guided vehicle systems. Int J Prod Res 35:943–953
46. Farling BE, Mosier CT, Mahmoodi F (2001) Analysis of automated guided vehicle configuration in flexible manufacturing systems. Int J Prod Res 39:4239–4260
47. Ross EA, Mahmoodi F, Mosier CT (1996) Tandem configuration automated guided vehicle systems: a comparative study. Decis Sci 27:81–102
48. Choi HG, Kwon HJ, Lee J (1994) Traditional and tandem AGV system layouts: a simulation study. Simul 63(2):85–93
49. Ho YC, Hsieh PF (2004) A machine-to-loop assignment and layout design methodology for tandem AGV systems with multiple-load vehicles. Int J Prod Res 42(4):801–832
50. Bartholdi JJ, Platzman LK (1989) Decentralized control of automated guided vehicles on a simple loop. IIE Trans 21 (1):76–81
51. Glover F (1989) Tabu search-Part I. ORSA J Comput 1:190–206
52. Glover F (1990) Tabu search-Part II. ORSA J Comput 2:4–32
53. Glover F, Taillard E, De Werra D (1993) A user's guide to tabu search. Annal Oper Res 41:3–28
54. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI
55. Seber GAF (1984) Multivariate observations. Wiley, New York
56. Roe A (1997) User's guide for LINDO and LINGO, Windows Versions. Duxbury Press, Belmont (Cal.)