IUST International Journal of Engineering Science, Vol. **19***, No.***1-2, 2008***, Page*** 33-42
Mathematics & Industrial Engineering Special Issue

A THRESHOLD ACCEPTING ALGORITHM FOR PARTITIONING MACHINES IN A TANDEM AUTOMATED GUIDED VEHICLE

R. Tavakkoli-Moghaddam, M.B. Aryanezhad, H. Kazemipoor and A. Salehipour

Abstract*: A tandem automated guided vehicle (AGV) system deals with grouping workstations into some non-overlapping zones, and assigning exactly one AGV to each zone. This paper presents a new non-linear integer mathematical model to group n machines into N loops that minimizes both inter and intra-loop flows simultaneously. Due to computational difficulties of exact methods in solving our proposed model, a threshold accepting (TA) algorithm is proposed. To show its efficiency, a number of instances generated randomly are solved by this proposed TA and then compared with the LINGO solver package employing the branch-and-bound (B/B) method. The related computational results show that our proposed TA dominates the exact algorithm when the size of instances grows.*

Keywords*: Tandem AGV, Machine Grouping, Mathematical Model, Threshold Accepting Algorithm.*

1. Introduction

Designing an efficient material handling system is a significant issue in the facility design. Tompkins et al. [1] showed that material handling costs are responsible for about 20 to 50 percent of the overall operational costs.

Following a path, an automated guided vehicle (AGV) is a driverless vehicle, which transports materials within a manufacturing area partitioned into cells. A tandem AGV problem was introduced by Bozer and Srinivasan [2-4] that is based on the divide-and-conquer principle.

They defined this system on a grid layout where 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.

In this paper, we consider the tandem AGV problem that partitions a set of *N* workstations into several independent, non-overlapping, single AGV, and closed loops (or zones) without any overlapping, in which there is exactly one AGV for each loop. Unlike traditional systems (Fig. 1), tandem layout is a mixed system consisting of two-way shortest path systems and one-way loops.

In such a system, all manufacturing areas are composed of several non-overlapping closed loops so that material flow in each loop is unidirectional. Additional pick-up/delivery (P/D) stations, in which bidirectional flow is possible (i.e. each pair of two loops is met), are introduced to provide an interface between adjacent loops (transition points).

It is clear that the number of required AGVs is equal to the total number of loops. Despite being defined and developed mainly for the manufacturing environment, tandem AGV systems can be used both in warehousing and manufacturing environments. Bozer and Srinivasan [2-3] stated the following advantages for tandem AGV systems:

- Simplifying control in each loop because of one AGV in each loop.
- Simplifying production processes in each loop.
- Removing collisions and traffic problems.
- Developing a group technology system.
- Finding optimal facilities' locations in each workstation.
- Increasing flexibility due to changes both in the number of workstations and in the production planning.

Paper first received December. 16, 2006 and in revised form April. 15, 2008.

R. Tavakkoli-Moghaddam: Associate professor in Department of Industrial Engineering and Engineering Optimization Research Group, University of Tehran, tavakoli@ut.ac.ir

M.B. Aryanezhad: professor in Department of Industrial Engineering, Iran University of Science and Technology, mirarya@iust.ac.ir

H. Kazemipoor: PhD student, Department of Industrial Engineering, Islamic Azad University-Parand Branch, hkazemipoor@yahoo.com

A. Salehipour: PhD student, Department of Industrial Engineering, Faculty of Engineering, Tarbiat Modarres University.

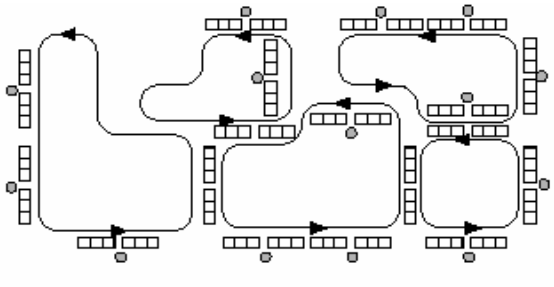

Fig 1. A tandem AGV system

In addition, there are some disadvantages including the need for handling a load by two or more vehicles, extra floor space, cost requirements, and the efficiency loss due to the transition points. In the tandem AGV system, workstations are partitioned in such a way that each station is assigned to only one loop. The workload of the AGVs associated with the material flow within and between the loops does not exceed the capacity of the AGV, and can be evenly distributed among all loops, if needed. As a building block of the system, the workload factor of each AGV, which is defined as the proportion of time the AGV is busy, either loaded or empty, should be calculated for each loop.

Bozer and Srinivasan [1-2] first proposed the tandem AGV problem, whose design issues have been the subject of studies by many researchers. Bozer and Srinivasan [3] studied partitioning of machines into different zones and developed the first heuristic to partition stations into loops. Faraji and Batta [5] pointed out some advantages and disadvantages of tandem AGV systems. Hsieh and Sha [6] modified an approximation solution for the *k*-TSP problem and developed a new algorithm for partitioning machines. Huang [7] proposed a new concept on design of the tandem AGV that links the transfer points using a transportation center. Choi et al [8] and Ross et al [9] conducted some experiments to compare a tandem AGV system with a conventional AGV system. They compared the performance measures of the two systems in various conditions, such as production, performance of vehicles, and mean production times. Aarab et al [10] designed tandem routes in a layout using hierarchical clustering and tabu search (TS) methods. They were first who developed a meta-heuristic method for the tandem AGV problem.

Gadmann and Velde [11] studied the problem of finding the best location of workstations in a loop using an AGV with *n* P/D points. Yu and Egbelu [12] presented a heuristic-based partitioning for the tandem AGV system based on the concept of variable path routing. Ventura and Lee [13] studied the tandem configurations with the possibility of using more than one AGV in each loop. Farling et al [14] carried out a simulation study to investigate the tandem configurations including the impact of system size, machine failure and unload/load time. Wooyeon and Egbelu [15] developed a partitioning heuristic for the problem based on variable path layout. They showed partitioning a tandem layout for an AGV system directly affects the operation of the system. Using simulation, they determined the location of transition points. Kim and Jae [16] analyzed the implementation of multi-load vehicles. Their algorithm has two objectives: 1) Minimizing the maximum workload of the AGV; and 2) minimizing the number of zones. Kim et al [17] proposed an analytical model in designing a tandem AGV system. At first, a traveling salesman problem (TSP) is solved to generate subsets of stations. The next step is to check the serviceability of the stations using the Markov chain model. The final tandem path is chosen using a partitioning algorithm. They used simulation to study the performance of the solution. Ho and Hsieh [18] considered balancing flow in each loop, minimizing inter-loop flow, and minimizing the distance of flow (i.e., the distance on which the flow has to travel) as three different objective functions. They first found a feasible solution for each loop using a heuristic method and then used a simulated annealing (SA) algorithm to improve the solution.

Fahmy, et al. [19] proposed a two-phase algorithm for partitioning machines with three objectives: 1) Minimizing total costs of material flow; 2) minimizing the maximum workload; and 3) minimizing the number of transitions in zones. They used shortest-time-to-travel-first (STTF) dispatching rule. They benefited from the simulation in demonstrating the effectiveness of their proposed algorithm. Shalaby, et al [20] developed the idea given in Fahmy, et al. [19] and proposed a two-phase algorithm to partition machines in AGV systems. In the first phase, the algorithm generates zones or feasible solutions and evaluates them. In the second phase, the algorithm uses an integer mathematical model to choose the best possible combination of zones.

On the path design issue of the tandem AGV, the most recent researches are due to Kaspi et al. [21], Ko and Egbelu [22], and Asef-Vaziri et al. [23] and [24]. Asef-Vaziri and Laporte [25] provided a complete survey on this problem.

According to the assumptions made by Bozer and Srinivasan [4], there are two types of workstations. The first type is the input/output station and the second type is the process station where the actual processing takes place. Transition points are considered as input/output (I/O) stations. When loaded, a bidirectional single load AGV is used in each loop and follows the shortest path to the destination station, and when empty it uses the first-encountered-first-served (FEFS) dispatching rule [26]. Additional assumptions are intersections and overlaps are avoided among loops. The number of loops must be at least two, which can be provided as input or through a design process. Solving the tandem AGV problem includes some sub-problems that are as follows: Assignment of machines into loops, decision on the number of required loops, and finally design of AGVs' routes. Depending on these sub-problems, different mathematical models should be solved. After modeling and solving these sub-problems, their solutions are combined together to form the original problem. This paper focuses on the first two objectives.

In this paper, a novel mathematical model of tandem AGV systems that partitions machines into *N* loops and proposes a threshold accepting (TA) algorithm as a solution procedure for this problem was presented. This new mathematical model and our proposed TA are the major contributions of this paper. The rest of this paper is organized as follows.

In Section 2, the problem and was formulated the notations used in the new mathematical model was explained. As for real life problems, due to computational difficulty, we could not find a good solution using exact algorithms in a reasonable amount of time, in the Section 3, a threshold accepting (TA) algorithm for solving the proposed model was designed. Section 4 contains experimental results of some randomly generated instances that were solved by both mathematical programming and the TA algorithm. Finally, this paper ends with the conclusion.

2. A Mathematical Model

In this section, the problem in a form of a non-linear mathematical model was formulated. It is worthy noting that the objective function was very similar to the cell formation problem given in [18]. Based on this idea and by using the dissimilarity coefficient proposed in [27], we presented a new non-linear mathematical model that determines a least cost assignment of machines to each loop, i.e. forms some loops with the minimum inter-loop and intra-loop flow. Following was the objective function:

$$\text{Min } Z = \{f_1(x), f_2(x), ..., f_N(x), g(x)\} \quad (1)$$

In this function, $f_j(x)$ s and $g(x)$ are singular objective functions in Z corresponding to intra and inter-loops, respectively, and N is the total number of loops. Note that Z is a multi-objective function, i.e. all $f_j(x)$ s and $g(x)$ have to be minimized where $f_i(x)$ s and $g(x)$ are as follows:

$$f_j(x) = \sum_{i=1}^{n}\sum_{\substack{k=1 \\ i\neq k}}^{n} f_{ik}x_{ij}x_{kj}, \quad j = 1,2,...,N \quad (2)$$

$$g(x) = \sum_{i=1}^{n}\sum_{\substack{j=1 \\ i\neq k}}^{N}\sum_{\substack{k=1 \\ j\neq l}}^{n}\sum_{l=1}^{N} f_{ik}x_{ij}x_{kl} \quad (3)$$

where, f_{ik} is flow between machines *i* and *k*, and x_{ij} takes 1 if machine *i* is assigned to loop *j*; and 0, otherwise.

It is clear to see that the given problem is to minimize a multi-objective function Z that is very difficult in general. One method to solve such a hard problem is to convert a multi-objective function problem into a singular objective function problem by removing some terms of the objective function and adding them as constraints. Thus, based on the balanced loop strategy, we remove parts of the objective function dealing with intra-loop flow ($f_j(x)$ s) and put them as constraints in our proposed model. The reason, which we use the balanced loop strategy, is that the model does not allow us to remove a machine or some machines. If we cannot do that, when the model is examining the different assignments, it may remove a machine from a loop and reassigns it to another loop. Although, this seems a feasible solution, but it reduces the flow in the loop whose machine is removed currently and increases the inter-loop flow. To avoid this, we used the balanced loop strategy. This helps to equalize both inter and intra-loop flow, simultaneously. Another important result of using the balanced loop strategy is to convert the multi-objective function Z into a singular objective function.

Finally, a brief discussion about the sensitivity analysis and the optimal number of total loops were provided in the last section of the paper. Now, we introduce decision variables and model parameters.

2.1. Decision Variables and Parameters

We define only one decision variable, x_{ij}, in the previous part informally. The formal definition of this variable is given bellow:

$$x_{ij} = \begin{cases} 1 & \text{if machine } i \text{ assigns to loop } j \\ 0 & \text{otherwise} \end{cases}$$

The model parameters are as follows:

f_{ik} Flow between machines *i* and *k*.

f_i Total flow from different machines to machine *i*, i.e., $f_i = \sum_{k=1}^{n} f_{ik}, \forall i$

T Total available time of AGV in the planning horizon in terms of time unit (total working time of AGV).

t_i Average pick-up, drop-off, and process times for one part on machine *i* in terms of time unit.

t'_j Bottleneck time in the *j*-th loop.

η, η' Lower and upper flow coefficients bounds in each loop.

N Pre-defined total number of loops.

2.2. Proposed Mathematical Model

Following is the mathematical model (i.e., Model (I)) for the above-mentioned problem.

Model (I):

$$\text{Min Z} = \sum_{i=1}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik}x_{ij}x_{kl}, \quad i \neq k, j \neq l \quad (4)$$

s.t.

$$\sum_{j=1}^{N} x_{ij} = 1 \quad , \quad i = 1,2,...,n \quad (5)$$

$$\sum_{i=1}^{n} x_{ij} \geq 2 \quad , \quad j = 1,2,...,N \quad (6)$$

$$\eta\left(\sum_{i=1}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik}x_{ij}x_{kl}\right) \leq \sum_{i=1}^{n}\sum_{k=1}^{N} f_{ik}x_{ij}x_{kl}, \quad j = 1,2,...,N \quad (7)$$

$$\eta'\left(\sum_{i=1}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik}x_{ij}x_{kl}\right) \geq \sum_{i=1}^{n}\sum_{k=1}^{N} f_{ik}x_{ij}x_{kl}, \quad j = 1,2,...,N \quad (8)$$

$$t'_j = \underset{i}{\text{Max}}\{f_i t_i x_{ij}\} \quad , \quad j = 1,2,...,N \tag{9}$$

$$\sum_{i=1}^{n} x_{ij} \le T / t'_j \quad , \quad j = 1,2,...,N \tag{10}$$

$$x_{ij} \in \{0,1\} \tag{11}$$

Equation (4) is the objective function that minimizes the total transitions between loops (inter-loop flow). This objective function calculates a cost incurred $j \neq l$, clearly this means in minimizing the inter-loop flow. Constraint (5) guarantees each machine must be assigned to only one loop. Constraint (6) assumes at least two machines should be assigned to each loop, however other values are possible. Constraints (7) and (8) emphasize a balanced flow in each loop. In these two constraints, η and η' show the flow ranges in loops and they are defined as follows:

$$\eta = 1/N - \Delta \tag{12}$$

$$\eta' = 1/N + \Delta \tag{13}$$

where, $0 \le \Delta < \frac{1}{N}$

In Eqs. (12) and (13), Δ represents the flow balance coefficient and is determined by the decision maker. Choices between small and large values for Δ correspond to more balanced and more unbalanced flow respectively.

If Δ takes small values, the range of flow changes in each loop will be small and hence the flow in the loops are closer to each other. On the other hand, if Δ takes large values, the flow in the loops are farther to each other. The choice of the upper bound for Δ avoids η' being negative and the choice of lower bound is optional and just makes a range for Δ changes. Before explaining constraints (9) and (10), we explain the following lemma.

Lemma 1: The capacity of each AGV to serve each loop has to be feasible as follows:

$$t'_j(\sum_{i=1}^{n} x_{ij}) \le T, \quad \forall j \tag{14}$$

Proof: We know the time requirement in each loop should be less than or equal to the total available time for each AGV in the planning horizon, thus:

$$f_1 t_1 x_{1j} + f_2 t_2 x_{2j} + ... + f_n t_n x_{nj} \le T, \ \forall j \tag{15}$$

It is worthy noting that in every operational process, the bottleneck machine or station determines the operation times and then it is required to calculate the process times based on that of bottleneck machine or station. It is easy to show the bottleneck time is calculated by constraints (9), i.e., $t'_j = \text{Max}(f_i t_i x_{ij})$. Furthermore, when $x_{ij} = 1$, t'_j holds its maximum value and we can replace all $f_i t_i$ by t'_j, thus we have Eq. (16).

$$t'_j(x_{1j} + x_{2j} + ... + x_{nj}) \le T \rightarrow t'_j(\sum_{i=1}^{n} x_{ij}) \le T, \ \forall j \tag{16}$$

This is the same as Constraint (10) and the proof is complete. Note that if $x_{ij} = 1$ then according to Constraint (5) loops exist, in which each holds at least two machines, so $t'_j \ge 0$.

Constraints (9) and (10) both test the feasibility of the solutions. In a case of infeasibility, the value of *N* has to be changed. Finally, the proposed mathematical model is a binary programming model and Constraint (11) imposes the binary restrictions on decision variables.

2.3. A Linear Model

The proposed mathematical model (i.e., objective function (4) and constraints (5) to (11)) is a non-linear model. As a fact, non-linear mathematical models are much more difficult to solve than the linear ones. Hopefully, it is easy to replace the non-linear objective function and some constraints with their corresponding linear ones, by using a general method in a quadratic assignment problem (QAP). First, we should add the following decision variable to our proposed model: $y_{ijkl} = x_{ij} x_{kl}$.

Adding variable y_{ijkl} requires some additional constraints (Constraints (17) to (19) given below). Clearly in the linear form, the objective function (4) changes into (4-1), we replace non-linear variables $x_{ij} x_{kl}$ with linear variable y_{ijkl} in Constraints (4), (7), and (8). Also, we replace Constraints (9) with their linear counterparts (9-1).

Model (II):

$$\text{Min Z} = \sum_{i=1}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik} y_{ijkl}, \ i \neq k, j \neq l, \tag{4-1}$$

$$x_{ij} + x_{kl} - 2y_{ijkl} \ge 0, \ \forall i, j, k, l \tag{17}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} y_{ijkl} = n(n-1)/2, \ k > i \tag{18}$$

$$y_{ijkl} \in \{0,1\}, \ \forall i, j, k, l \tag{19}$$

$$t'_j \ge (f_i t_i x_{ij}), \ \forall i, j, k, l \tag{9-1}$$

$$\sum_{i=1}^{n} x_{ij} \le T / t'_j, \ j = 1,2,...,N \tag{10}$$

$$y_{ijkl}, \ x_{ij} \in \{0,1\} \tag{20}$$

where, $y_{ijkl} = \begin{cases} 1 & \text{if both } x_{ij} \text{ and } x_{kl} \text{ are 1} \\ 0 & \text{Otherwise} \end{cases}$

We call this new model with the linear constraint, Model (I) (i.e., objective function (4-1) with Constraints (17) to (19), (9-1), (10), and (20)). As it can be seen, Constraints

(10) are still non-linear. However, clearly Model (II) is much easier to solve than Model (I). As we shall see later in computational results section, we use this simple mathematical model to solve small instances to find a robust comparison way with the result of our proposed threshold accepting (TA) algorithm.

3. An Efficient Meta-Heuristic Algorithm

By looking into our mathematical model and its objective function, it is not difficult to see that the objective function is the well-known QAP model. According to this and also taking into account the constraints, it can be stated this combinatorial optimization problem is an NP-Hard one. We know that the QAP model is also an NP-Hard problem [27]. Difficulty of NP-Hardness has made researchers to focus on meta-heuristic. In this paper, we use threshold accepting (TA) algorithm [28] to find high quality and promising solutions for the above hard problem. TA is one of well-known meta-heuristics for obtaining good and near-optimal solutions to difficult optimization problems, which has received much attention over the last few years. The TA algorithm is a type of a local search algorithm and avoids becoming trapped in a local optimum by sometimes accepting a move that makes worse the objective function value (OFV). Simply, it means that the TA algorithm also accepts the worse solutions. Denoted by τ, the acceptance or rejection of a move is determined by this threshold value, which is decreased during the algorithm. This acceptance criterion implies that when algorithm begins and the threshold value τ has its maximum value to avoid being trapped in a local optimum, large variations in the objective function f are more often accepted. As the algorithm progresses and the τ decreases only small variation in the objective function f are accepted. By variation in the objective function, f, we mean deviation from the OFV that makes worse the OFV. More details about TA can be found in [28].

In fact, the TA is the deterministic variant of the simulated annealing (SA) algorithm. While the SA accepts a worst move based on a probability, the TA algorithm accepts the worse move based on the pre-determined threshold. Thus, except this threshold, all steps are the same as the SA algorithm. To implement the TA algorithm, the following steps have to be considered:

- Generating a feasible initial solution,
- Defining a set of neighborhoods to produce neighbor solutions (i.e., local searches),
- Setting a stopping condition,
- Tuning the threshold value τ,

Our proposed TA algorithm works as follows: Given an initial solution, it selects one of the three neighborhoods randomly, where neighborhoods are thoroughly explained later in this section, to search the solution space. In fact, these neighborhoods perform as local search algorithms. The result from applying these neighborhoods is the current solution or incumbent solution that has to be evaluated. If this incumbent solution is better than the best found solution so far, it will be accepted by the algorithm, otherwise it will be accepted according if its difference with the best found solution so far (i.e., incumbent solution does not exceed the threshold value τ). The algorithm rejects the solution if this difference is beyond the threshold value, τ. This process is repeated until stopping condition is met. Fig. 2 shows a pseudo-code version of our proposed TA algorithm.

Begin
Generate an initial solution x and let it be the incumbent solution;
Set $r = 0$ and three neighborhoods $N_k(x)$ (for k=1, 2, 3), also initialized $R_{\max}$ and τ;
While there is improvement **Repeat**
  Set k=1, i.e. search using the first neighborhood, set τ to its initialized value;
  For $r = 1$ to $r = R_{\max}$ **Do**
    While stopping condition is not met **Do**
      Generate a neighbor solution of x using the k-th neighborhood. Let this solution be x';
      If $f(x') < f(x)$ ($f(x') > f(x)$) for a minimization (maximization) problem, then accept x' and let $x := x'$
      Otherwise calculate $\Delta = f(x') - f(x)$ ($\Delta = f(x) - f(x')$) for a minimization (maximization) problem;
      If $\Delta < \tau$, then accept x' and let $x := x'$;
    End While
    Reduce τ;
  End For
  k=k+1;
End

Fig 2. Pseudo-code version of our proposed TA

3.1. Generating a Feasible Solution

Like every improvement local searches, the TA depends on the quality of an initial solution. Before generating an initial solution, we modify the objective function to satisfy all constraints. To do this, we implement the Lagrangian relaxation (LR) approach [29]. In this approach, those constraints that are difficult to be satisfied are removed and considered in the objective function with a penalty. To clarify this, let us define the following mathematical model with the objective function C and constraints $g_i(x)$s:

$$\text{Min. } C = f(x)$$
$$\text{S.T.} \qquad (21)$$
$$g_i(x) \le b_i, \forall i$$

Since the constraints are "less-than-equal-to" constraints, a penalty function given in [28] was used to put the constraints in the objective function:

$\text{Min } \{f(x) + rp(x)\}$ where, $p(x)$ is the penalty function and,

$$p(x)=\sum_{i=1}^{m}\text{Max }\{(g_i(x)-b_i),0\}^2 \qquad (22)$$

Now, coincidently the problem (21) be equal to the problem (22), then in our mathematical model, the new objective function after transformation using the penalty function can be define as follows:

Model (III):

$$\text{Min}\Big(\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq l}}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik}X_{ij}X_{kl}+r\sum_{j=1}^{N}\Big($$

$$\text{Max}\left[\left\{\eta\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik}X_{ij}X_{kl}-\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{k=1}^{n} f_{ik}X_{ij}X_{kj}\right\},0\right]$$

$$+\text{Max}\left[\left\{\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{k=1}^{n} f_{ik}X_{ij}X_{kj}-\eta'\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} f_{ik}X_{ij}X_{kl}\right\},0\right]^2$$

$$+\text{Max}\left[\left\{\sum_{i=1}^{n}X_{ij}-T*1/t'_j\right\},0\right]^2\Big) \qquad (23)$$

Objective function (23) consisting of the penalty functions, subject to constraints (5), (6), (9), (11), and (19), forms Model (III) and this model is equal to Model (II). Using Model (III), one can state each random assignment of machines to loops while considering at least two machines in each loop and calculating t'_j by constraints (7) or (7-1). This results in finding a feasible solution for Model (III). To generate a good initial solution for Model (III), we use the modified spanning tree (MST) concept illustrated in [11] with a little change as can be depicted in Fig. 3.

3.2. Neighborhood Structures and Stopping Condition

After generating an initial solution, the next step was to improve that solution. We did this by three neighborhood structures or local searches. These three searches were remove-insert neighborhoods and they differed in the way they choose a loop to remove a machine from it and in the way they choose a machine to be removed.

The first neighborhood choosed the loop that had the maximum flow and the second choosed the loop that had the maximum number of machines. The third neighborhood was a probabilistic neighborhood, in which machines were chosen to be removed according to a probability proportional to the ratio of their selection in the two previous neighborhoods. Following is the definition of these three neighborhood structures, namely N_1, N_2, and N_3.

- Maximum flow remove-insert neighborhood (N_1): Given a solution (i.e., an incumbent solution), this neighborhood calculates the total flow in each loop, finds the maximum flow, then removes a machine randomly from the selected loop and reassigns it to the loop that has the minimum flow.

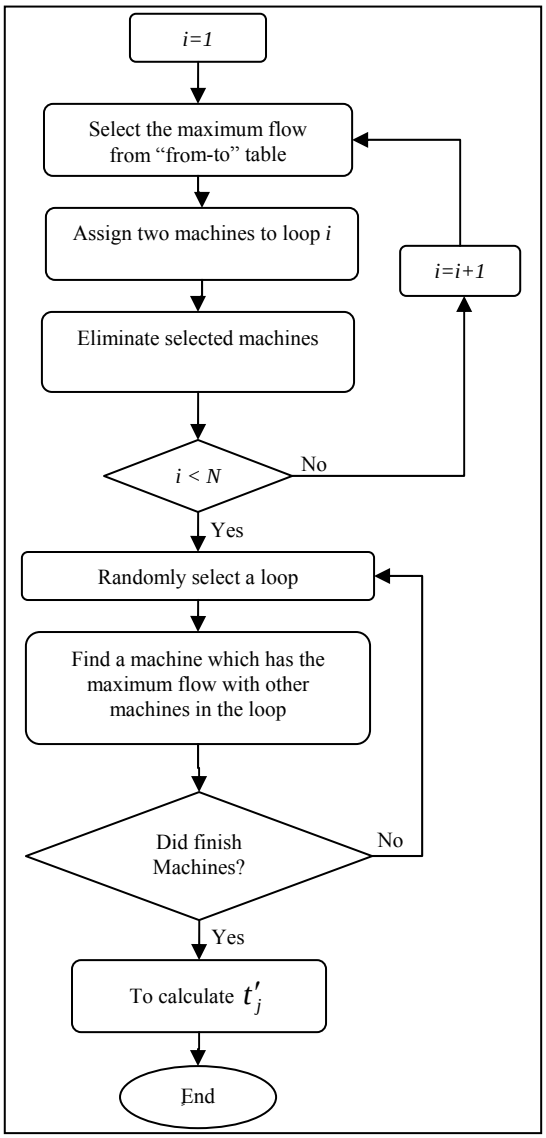

Fig 3. Process of generating an initial solution

- Maximum machine number remove-insert neighborhood (N_2): Given a solution, this neighborhood chooses a loop that has the maximum number of machines, removes a machine randomly from it and reassigns it to a loop that has the minimum number of machines.
- Probabilistic remove-insert machine neighborhood (N_3): Given a solution, this neighborhood calculates the probability of selecting a machine in the N_1 and N_2 neighborhoods and then chooses a machine to be removed from a loop according to this probability and reinserts it in its best position among other loops.

At each iteration, our proposed TA algorithm moves between these three neighborhoods randomly, implements one of them, finds an incumbent solution and then calculates the objective function for this incumbent solution. If the objective function value for the incumbent solution is better than the objective function value for the best solution found so far, the algorithm replaces the best found solution, otherwise the algorithm accepts this worsening move if deteriorated OFV is less than the threshold value τ. In either case, the algorithm decreases the τ and goes to the next iteration. To terminate the algorithm, we set stopping criterion as the value of 0 for τ.

4. Computational Results

To demonstrate the performance of our proposed algorithm, we tested our algorithm on 5 problems which their "from-to" tables are generated randomly using uniform distribution between [0,10] and [0,100] ($U \sim (0,10)$ and $U \sim (0,100)$). We use two intervals for generating the numbers to demonstrate the final solution is not dependent on some specific values.

One of the main differences between mathematical models and meta-heuristics lies in using the parameters. Usually in mathematical models, the results are less dependent on these parameters' values, as mathematical models are considered exact methods. However due to high computational time, real-life problems cannot be solved by these exact methods.

On the other hand, meta-heuristics are more dependent on those parameters since they are not exact and changing those parameters can affect the quality and time requirement of the final solution. In fact, these parameters guide the search towards promising directions. For tuning our TA only parameter, τ, we use some small-sized problems that are solvable by using our mathematical model.

Then, we tuned the τ in a way that the TA generates the best and closest solutions to the solution of our mathematical model. Experiments showed the initial value of 100 with a decreasing step of 0.1 was a good choice.

Parameters for the proposed mathematical model are brought in Table 1. Table 2 shows the results of our proposed TA algorithm and the associated results of the LINGO solver that is an exact solver. Also in Table 2, readers can compare the solution by LINGO, TA algorithm and the SA algorithm very easily. The SA results are from [30].

As can be seen by this table, in most cases our proposed TA algorithm has found good solutions very close to LINGO solver and in some cases even better than LINGO in much less computational time (i.e., it means those solutions for LINGO are not global optimal). Note that for small-sized problems, one can use exact solvers like LINGO to find the solution and for large-sized and real-life problems exact solvers are unable to find the solution either global or local optima in a reasonable amount of time (i.e., polynomial time). The error in Table 2 is calculated using the results of the LINGO solver and the TA algorithm.

These small-sized problems are the proof of the accuracy of our mathematical model. The results obtained by the proposed TA algorithm are best results found in all 15 runs.

In Figure 5, we present the average and variance of the OFV for our five benchmark problems considering all ten runs of our proposed TA algorithm. Table 3 shows data for 5 medium and 15 large-sized problems. In this table, except problem 11 taken from [6], all other problems were generated randomly between [0,100] (i.e., U~(0,100)). Table 4 shows computational results for the problem instances illustrated in Table 3.

In Table 4, we also report the computational result of the SA algorithm proposed in [30]. To keep the model easy, the value for parameter t_i for problems 6 to 12 was chosen 1.

Tab. 1. Parameters' values for five instances (P: Problem, M: Number of machines, L: Number of loops)

Instances	Parameters			
	η	η'	t_i	T
P 1 (5M, 2L)	0.15	0.85	{3,1,2,4,1}	4000
P 2 (6M, 2L)	0.25	0.75	{1,1,1,1,1,1}	2000
P 3 (7M, 2L)	0.2	0.8	{2,1,5,1,3,1,1}	1000
P 4 (8M, 2L)	0.25	0.75	{2,1,5,1,3,1,1,3}	4000
P 5 (9M, 2L)	0.2	0.8	{2,1,5,1,3,1,1,3,4}	5000

Tab. 2. Computational results of our proposed TA algorithm and LINGO (PC: Windows XP, RAM 1024 MB, CPU 3 GHz)

Instances	LINGO		TA		SA		% Deviation from LINGO
	OFV	CPU time (s)	OFV	CPU time (s)	OFV	CPU time (s)	
P1	**547**	7	547	1.45	547	1.62	0
P2	**552**	35	552	2	573	2.12	0
P3	**80**	230	80	2.8	80	3	0
P4	**118**	1847	121	2.8	121	3.31	2.48
P5	**1694**	25586	1715	3.12	1710	3.52	0.94

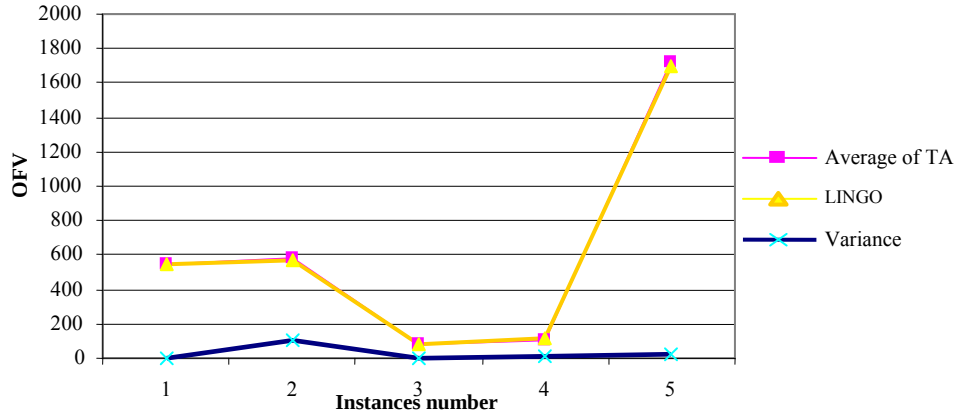

Fig 4. Analysis for five problems

Tab. 3. Parameter values (P: Problem, M: Number of machines, L: Number of loops)

Instances	Parameters			
	η	η′	t_i , $\forall i$	T
P6 (10M, 2L)	0.15	0.85	1	5000
P7 (11M, 2L)	0.2	0.8	1	5000
P8 (12M, 3L)	0.09	0.57	1	5000
P9 (15M, 3L)	0.09	0.57	1	5000
P10 (20M, 4L)	0.05	0.45	1	5000
P11 (24M, 4L)	0.05	0.45	1	5000
P12 (30M, 5L)	0.02	0.38	1	5000
P13 (30M, 5L)	0.02	0.38	1	5000
P14 (40M, 5L)	0.02	0.38	1	5000
P15 (40M, 5L)	0.02	0.38	1	5000

In [6] by using a grouping method, problem 11 had the OFV of 2400, which had been improved by our TA algorithm to 2100 (i.e., 12.5%). Considering other parameters given in [6], it is clear to see that in problem 11, the results found by our TA algorithm decreased the total distances of the loop. Finally, as shown in Table 4, the LINGO solver after 10 hours of running just generated feasible solutions for problems 6 and 7, and not even the feasible solutions for problems 8 to 15. However, our TA algorithm can find the local optima very quickly. To best of our knowledge, except problem 11, for other instances, there are no optimal solutions or best found solutions yet to be able us to evaluate our results.

Tab. 4. Computational results of the TA algorithm and LINGO (PC: Windows XP, RAM 1024 MB, CPU 3 GHz)

Instances	Best OFV from LINGO after 10 hours	TA algorithm		SA algorithm	
		OFV	CPU time (sec.)	OFV	CPU time (sec.)
P6	2194	2158	5.45	2158	5.81
P7	3278	3127	12	3127	12.81
P8	-	4765	14.5	4765	15.79
P9	-	**6632**	15.25	6664	17.23
P10	-	**15764**	19.8	15776	20.16
P11	-	2100	20.2	2100	20.28
P12	-	**25500**	20.5	25598	21.35
P13	-	**24645**	22	24640	24.56
P14	-	**32112**	36	32220	42.44
P15	-	**34256**	34.95	35008	45.35

However, by five previous small-sized problems as well as problem 11, we can conclude the effectiveness and robustness of our proposed TA algorithm.

Throughout the paper, we assume that the value for N (i.e., the total number of loops) is known in advance. However, finding its optimum value can be attained in a simple way: the idea is to solve the problem instances by using mathematical Mode (l) and in case of large-sizes or real-life instances by using the TA algorithm, for different values of N and then choose the value of N for which the objective function is minimum or optima.

5. Conclusion

In this paper, we have proposed a novel, non-linear mathematical model to partition machines into loops in tandem AGV systems.

This proposed model has bi-objectives minimizing inter and intra-loop flows. The purpose of introducing the nonlinear model is to formulate the problem based on its nature; however, since it is so difficult to solve this non-linear model, by converting the non-linear objective function and some constraints into linear ones, it is going to be easier to solve the model.

However, due to the complexity of our new mathematical model, namely Model (II), a threshold accepting (TA) algorithm has been developed and implemented to solve the given problems.

Using different problem instances, we have compared the computational results based on both our proposed TA algorithm and the LINGO solver, as a well-known optimization solver. These results show the high convergence rate of our proposed TA algorithm and also high quality solutions. For future research, using other meta-heuristics can be proposed. Also developing other powerful neighborhood structures to be chosen in a systematic and structured way (e.g., variable neighborhood search (VNS) algorithm) can be considered for further research.

References

[1] Tompkins, J.A., White, J.A., Bozer, Y.A. Tanchoco, J.M.A., "*Facilities Planning, 3rd Edition*", John Wiley & Sons, New York, 2003.

[2] Bozer, Y.A., Srinivasan, M.M., "*Tandem Configurations for Automated Guided Vehicle Systems Offer Simplicity and Flexibility"*, Industrial Engineering, Vol. 21, No. 2, 1989, PP. 23-27.

[3] Bozer, Y.A., Srinivasan, M.M., "*Tandem Configurations for Automated Guided Vehicle Systems and the Analysis of Single-Vehicle Loops"*, IIE Transactions, Vol. 23, No. 1, 1991, PP. 72-82.

[4] Bozer, Y.A., Srinivasan, M.M., "*Tandem AGV Systems: A Partitioning Algorithm and Performance Comparison with Conventional AGV Systems"*, European Journal of Operational Research, Vol. 63, No. 2, 1992, PP. 173-192.

[5] Faraji, M., Batta, R., "*Forming Cells to Eliminate Vehicle Interference and System Locking in an AG*VS", International Journal of Production Research, Vol. 32, No. 9, 1994, PP. 2219-2241.

[6] Hsieh, L.F., Sha, D.Y., "*A Design Process for AGV Systems",* Integrated Manufacturing System, Vol. 7, No. 6, 1996, PP. 30-38.

[7] Huang, C., "*Design of Material Transportation System for Tandem Automated Guided Vehicle Systems"*, International Journal of Production Research, Vol. 35, 1997, PP. 943-953.

[8] Choi, H.G., Kwon, H.J., Lee, J., "*Traditional and Tandem AGV System Layouts: A Simulation Study*," Simulation, Vol. 63, No. 2, 1994, PP. 85-93.

[9] Ross, E.A., Mahmoodi, F., Mosier, C.T., "*Tandem Configuration Automated Guided Vehicle Systems: A Comparative Study"*, Decision Sciences, 27, 1996, PP. 81-102.

[10] Aarab, A., Chetto, H., Radouance, L., "*Flow Path Design for AGV Systems"*, Studies in Informatics and Control, 8(2), 1999, PP. 97-106.

[11] Gadmann, A.G.R.M., Velde, S.L., "*Positioning AGV in a Loop Layout"*, European Journal of Operation Research, Vol. 127, 2000, PP. 565-573.

[12] Yu, W., Egbelu, P., "*Design of a Variable Path Tandem Layout for Automated Guided Vehicle Systems"*, Journal of Manufacturing Systems, Vol. 20, 2001, PP. 305-319.

[13] Ventura, J.A., Lee, C.U., "*Tandem Loop with Multiple Vehicles Configuration for Automated Guided Vehicle Systems"*, Journal of Manufacturing Systems, Vol. 20, 2001, PP. 153-165.

[14] Farling, B.E., Mosier, C.T., Mahmoodi, F., "*Analysis of Automated Guided Vehicle Configuration in Flexible Manufacturing Systems",* International Journal of Production Research, Vol. 39, 2001, PP. 4239-4260.

[15] Wooyeon, Y., Egbelu, P.J., "*Design of Variable Path Tandem Layout for AGV",* Journal of Manufacturing System, Vol. 20, No. 5, 2001, PP. 305-319.

[16] Kim, K.S., Jae, M., "*A Design for a Tandem AGVS with Multi-Load AGVs",* International Journal of Advanced Manufacturing Technology , Vol. 22, 2003, PP. 744 -752.

[17] Kim, K.S., Chung, B.D., Jae, M., "*A Design for a Tandem AGVS with Multi-Load AGVs"*, International Journal of Advanced Manufacturing Technology, Vol. 22, 2003, PP. 744-752.

[18] Ho, Y.C., Hsieh, P.F., "*A Machine-to-Loop Assignment Layout Methodology for Tandem AGVS with Multiple-Load Vehicle*," International Journal of Production Research, Vol. 42, No. 4, 2004, PP. 801-832.

[19] Fahmy, S.A., Shalaby, M.A., Elmekkawy, T.Y., "*A Multi-Objective Partitioning Algorithm for Tandem AGV System"*, 35th International Conference on Computers and Industrial Engineering, 2005, PP. 715-720.

[20] Shalaby, M.A., Elmekkawy, T.Y., Fahmy, S.A., *"A Cost Based Evaluation of a Zones Formation Algorithm in Tandem AGV Systems",* International Journal of Advanced Manufacturing, DOI: 10.1007/s00170-005-0163-1, 2005.

[21] Kaspi, M., Kesselman, U., Tanchoco, J.M.A., "*Optimal Solution for the Flow Path Design Problem of a Balanced Unidirectional AGV System*," International Journal of Production Research, Vol. 40, 2002, PP. 349- 401.

[22] Ko, K.C., Egbelu, P.J., *Unidirectional AGV Guide Path Network Design: A Heuristic Algorithm*, International Journal of Production Research, Vol. 41, 2003, PP. 2325-2343.

[23] Asef-Vaziri, A., Laporte, G., Sriskandarajah, C., *"The Block Layout Shortest Loop Design Problem".* IIE Transactions, Vol. 32, 2000, PP. 724-734.

[24] Asef-Vaziri, A., Dessouky, M., Sriskandarajah, C., A *"Loop Material Flow System Design for Automated Guided Vehicles*", International Journal of flexible Manufacturing Systems, Vol. 13, 2001, PP. 33-48.

[25] Asef-Vaziri, A., Laporte, L., Ortiz, R., "*Exact and heuristic procedures for the material handling circular flow path design problem"*, European Journal of Operational Research, Vol. 176, No. 2, 2007, PP. 707-716.

[26] Bartholdi, J.J., Platzman, L.K., "*Decentralized Control of Automated Guided Vehicles on a Simple Loop*," IIE Transactions, Vol. 21, No. 1, 1989, PP. 76-81.

[27] Heragu, S.S., "*Facilities Design"*, PWS Publishing Company, 1997.

[28] Dueck, G., Scheuer, T., "*Threshold Accepting. A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing"*, Journal of Computational Physics, Vol. 90, 1990, PP. 161-175.

[29] Bazaraa, M.S., Sheraly, H.D., "*Nonlinear Programming"*, John Wiley, 2nd edition, NY, 1993.