A scheduling Approach for the design of Flexible Manufacturing Systems

Laurent Deroussi, Michel Gourgand

A scheduling Approach for the design of Flexible Manufacturing Systems

L. DEROUSSI[1], M. GOURGAND[2]

LIMOS CNRS UMR 6158
Université Blaise Pascal, Clermont-Ferrand
[1] IUT d'Allier, 03101 Montluçon, France
[2] ISIMA, Campus des Cézeaux, 63173 Aubière, France

Corresponding author, Email: deroussi@moniut.univ-bpclermont.fr, Tel: (33) 4.70.02.20.97

Abstract The aim of this chapter is to show the interest of considering scheduling problems for the design improvement of Flexible Manufacturing Systems (FMS). The problem occurs when we wish to reconsider the design of an FMS by evaluating its functioning using a low-level model. The experimental results we have obtained show that, in many cases, a simple reorganization of the production cells can improve the overall productivity of an FMS. This problem is known in the literature as the machine assignment problem. The low-level evaluation model we have considered permits to synchronize the handling material system with the production tools, in order to minimize the time required for the production of a given set of jobs (makespan). We propose a hybridized solution approach that combines different optimization strategies: integer linear programming, metaheuristics and discrete event approaches. Integer linear programming is used for elaborating an initial machine assignment (an optimal one for the high-level model). New machine assignments are constructed by ants, and evaluating thanks to a low-level model evaluation. The latter is a black box optimization subroutine that combines iterated local search and discrete event approach.

Keywords: Flexible Manufacturing Systems, Scheduling Problem, Machine Assignment Problem, Hybrid methods.

1. INTRODUCTION

Flexible Manufacturing systems (FMS) are highly automated production systems in which a set of numerically-controlled machine tools is linked together by a material handling system. One of the most popular handling system is *automated guided vehicles* (AGV). FMS become more and more popular because of their high productivity and flexibility. On the other hand, FMS are expansive to build and complicated to design. As explained by Bhattacharya *et al*. (2008), the process design of an FMS consists of a set of crucial

decisions that have to be made carefully. FMS have been studied by numerous researchers during the last decades. These studies can be classified into three main categories: facility layout problem, guide-path design and vehicle scheduling.

The facility layout problem (FLP) concerns the placement of the facilities into the plant area in order to minimize the cost of total material flow between facilities. This problem is usually modelled by a quadratic assignment problem (QAP) (see (Drira *et al.*, 2006) for a survey). For a given facility layout, the guide path design includes the flow-path layout, the location of the pick up and drop areas (P/D), and the fleet size. Lastly, vehicle scheduling concerns the scheduling of jobs on the machines, the AGV dispatching rules and the solution of conflicts between AGVs. We can distinguish offline scheduling and online scheduling. In the offline situation, all transportation requests are assumed to be known in advance. The vehicle routes between machines are calculated offline before the vehicles take them. In the online situation, the path between two machines is dynamically determined as a function of the state of the system. (Ganasharajah *et al.*, 1998; Le Anh & De Koster, 2006) propose comprehensive surveys of the two last topics.

Although design problems are closely related, they have traditionally been tackled separately because of their computational intractability (Kim & Goetschalckx, 2005). The authors propose a heuristic for the concurrent determination of the block layout, the P/D points and the guide path between P/D points, but they only consider the minimization of the loaded vehicle travel distance. Several authors have noticed the relevance of minimizing the total vehicle travel distance (Sun & Tchernev, 1996; Ganasharajah *et al.*, 1998; Asez-Vaziri *et al.*, 2007). In the latter, the authors claim that “the main costs of an AGV system are the investment cost in the fleet of the vehicles, the electricity consumption, and the wasted material handling capacity due to AGV battery re-charging times. These are all functions of the total required loaded and empty travel distances”. In (Asez-Vaziri *et al.*, 2007; 2008) the authors consider the empty vehicle travels for designing flow path and P/D location in the special case of unidirectional loop layout. They conclude their study by noting that the ignorance of empty travel may lead to the design of layouts that are far from optimal.

In this chapter, we consider the machine layout problem in FMS environment of type job shop or flexible job shop. The problem is to assign machines to locations within a given layout arrangement such that a given performance measure is optimized. Several authors (El Baz, 2004; Ray & Sarker, 2008) work on the machine assignment problem for different categories of layouts (flow-line, multi-line, closed loop). The first paper focuses on the machine assignment problem, while the two others include also the job assignment problem. Nevertheless, all these authors minimize the total costs of the material flow (the sum of transport costs and machining costs for the two last papers) and they propose evolutionary algorithms to solve their problem. The objective pursued in this paper is different. We want to show that the solution obtained by solving the design problem (tactical level or high-level model) is not necessarily the best solution when we consider scheduling problems (operational level or low-level model).

The remainder of this chapter is organized as follows. In the first section, we detail the studied problem and we introduce some notations. The next section gives a description of the proposed solution approach. This approach is applied to FMS of job shop type and flexible job shop type.

Finally, we conclude and propose some perspectives.

2. DESCRIPTION OF THE PROBLEM AND NOTATIONS

In this section, we give more details about the studied problem and we introduce the notations used in the remainder of the chapter.

2.1. Problem definition

The problem under consideration consists in the simultaneous solution of machine layout problem, scheduling of AGVs and scheduling of machines in an FMS environment. We consider a given layout arrangement where the entire shop floor is divided into a number of locations. We assume that all the locations are of equal area. Any machine could be assigned to any location, with no compatibility restriction. These locations are linked together by a guide path network, as shown in figure 1. The arrows indicate if the arcs are unidirectional or bidirectional. The value on each arc indicates the travel time (or distance) when a vehicle moves along the section. Table 1 gives the shortest path between any two locations. The FMS is of flexible job shop type. Each job consists of a set of operations that must be processed in the given order. Each operation needs to be processed during a given amount of time on a given type of machines. A type of machine characterizes a subset of the machines (identical or not). Each machine can handle at most one operation at a time. Each job has to be transported from an origin machine to a destination machine by AGV. The problem is to determine the assignment of machines to locations, which minimizes the completion time of a given set of jobs (makespan). The calculation of the makespan requires synchronization between the production tools and the material handling system. Each of these two problems, when examined separately, is NP-hard. It is, according to the best of our knowledge, the first time that the makespan criterion is used to tackle design problems in FMS environments.

If the FMS is of job shop type, each machine is the single representative of its type. It is a special case of the FMS flexible job shop problem, for which an operation could be achieved by several machines. Thus, it will raise the question of the choice of the machine that will achieve this operation. Finally, we won't consider in this chapter the FMS flow shop scheduling. Indeed, for this kind of FMS, the guide path layout is often designed with an inner loop. In this case, the machine assignment problem is less relevant.

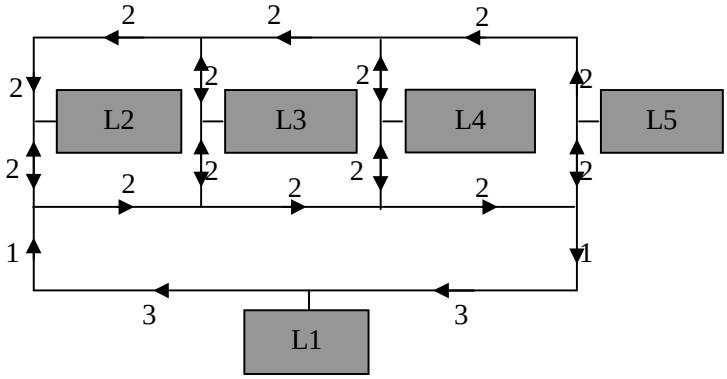

Fig. 1. *An illustration of a layout with 5 locations[1] and its related travel time matrix.*

Table 1. *Shortest path travel times matrix between locations.*

	L1	L2	L3	L4	L5
L1	0	6	8	10	12
L2	12	0	6	8	10
L3	10	6	0	6	8
L4	8	8	6	0	6
L5	6	10	8	6	0

2.2. Notations

In the remainder of this paper, we will use the following notations:

- $J = \{J_1, \cdots, J_{\bar{n}}\}$ the set of $\bar{n}$ independent jobs,
- $L = \{L_1, \cdots, L_{\bar{m}}\}$ the set of $\bar{m}$ locations,
- $T = \{T_1, \cdots, T_{\bar{t}}\}$ the set of $\bar{t}$ types of machine,
- $M = \{M_1, \cdots, M_{\bar{m}}\}$ the set of $\bar{m}$ machines (including the load / unload machines),
- $m_t, t \in [\![1, \bar{t}]\!]$ the number of machines of type T_t,
- $V = \{V_1, \cdots, V_{\bar{k}}\}$ the set of $\bar{k}$ vehicles,
- α_j the number of operations of job $J_j \in J$,
- $O^j = \{o_{ji}, i \in [\![1, \alpha_j]\!]\}$ the set of operations of job $J_j \in J$,
- o_{j0} a fictitious load operation of job $J_j \in J$
- $O = \bigcup_{J_j \in J} O^j$ the set of all the operations which must be performed,
- $O^+ = \bigcup_{J_j \in J} \left(O^j \cup \{o_{j0}\}\right)$ the set of all operations plus the fictitious ones,
- $\left(\mu_{ji}, \left(p_{ji1}, ..., p_{ji,m_{\mu_{ji}}}\right)\right) \in \left([\![1, \bar{t}]\!] \times \mathbb{N}^{m_{\mu_{ji}}}\right)$, respectively the type $T_{\mu_{ji}}$ of machine and the $m_{\mu_{ji}}$-tuple of processing times required for the operation $o_{ji} \in O^+$. The i^{th} value in the tuple corresponds to the i^{th} machine of type $T_{\mu_{ji}}$ given by the natural order of the machines. By convention, μ_{j0} is the type of the loaded stations and $p_{j0m} = 0$ for $m \in [\![1, m_{\mu_{j0}}]\!]$.
- $\tau_{mt} = \begin{cases} 1 \text{ if machine } M_m \text{ is of type } T_t \\ 0 \text{ otherwise} \end{cases}$
- $\left(t_{l_1 l_2}, t'_{l_1 l_2}\right) \in \mathbb{N}^2$ respectively the loaded and empty travel times between locations l_1 and l_2.

In the important peculiar case of identical machines for each type, these notations can be simplified by replacing $\left(\mu_{ji}, \left(p_{ji1}, ..., p_{ji,m_{\mu_{ji}}}\right)\right) \in \left([\![1, \bar{t}]\!] \times \mathbb{N}^{m_{\mu_{ji}}}\right)$ with $\left(\mu_{ji}, p_{ji}\right) \in \left([\![1, \bar{t}]\!] \times \mathbb{N}\right)$.

3. THE SOLUTION APPROACH

We present in this section the proposed solution approach. The general framework is summarized in figure 2. We propose first a formulation of our problem for the minimization of loaded vehicle travels as a quadratic assignment problem (QAP). The machine assignment obtained by the solution of this QAP is used to define a TSP-like problem on which we will apply Ant Colony System. The goal of ACS is to build diversified machine assignments, which are correlated to the initial assignment. These randomly constructed machine assignments are evaluated using a black-box optimization algorithm derived from a previous work (Deroussi *et al.*, 2008).

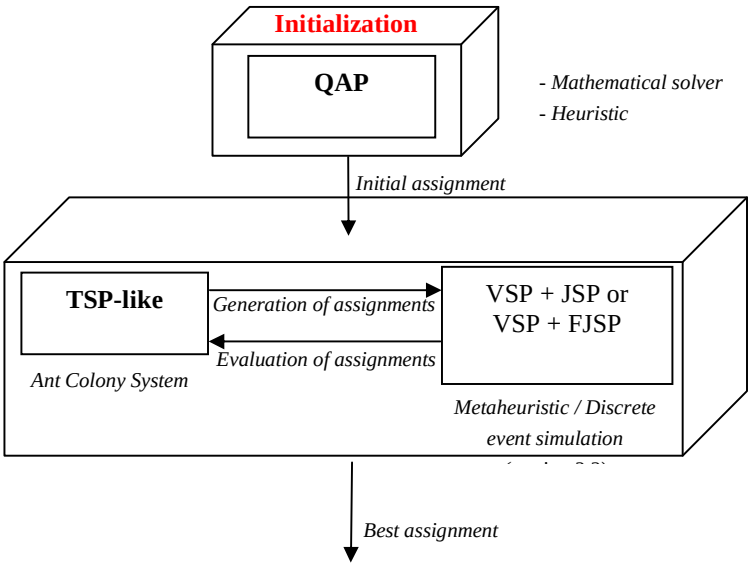

Fig. 2. *The proposed solution approach.*

3.1. Determination of the initial solution

The location of machines can be determined by minimizing the total loaded vehicle travel distance. The general case of the FMS flexible job shop scheduling problem is first investigated. A quadratic 0-1 integer formulation is proposed for this problem. The peculiar case of the FMS job shop scheduling problem is then considered as a simplification of the general case.

Quadratic 0-1 integer formulation for the FMS flexible job shop

The problem of finding the machine assignment that minimizes the total loaded vehicle travel distance, in the case of FMS flexible job shop problem, can be formulated as follows:

$$\text{Minimize} \quad \sum_{o_{ji} \in O} \sum_{l_1=1}^{\bar{m}} \sum_{l_2=1}^{\bar{m}} t_{l_1 l_2} y_{o_{j,i-1} l_1} y_{o_{ji} l_2} \qquad (1)$$

$$\text{Subject to} \quad \sum_{m=1}^{\bar{m}} x_{ml} = 1, \; l \in [\![1, \bar{m}]\!] \qquad (2)$$

$$\sum_{l=1}^{\bar{m}} x_{ml} = 1, \; m \in [\![1, \bar{m}]\!] \qquad (3)$$

$$\sum_{l=1}^{\bar{m}} y_{o_{ji} l} = 1, \; o_{ji} \in O^{+} \qquad (4)$$

$$y_{o_{ji} l} \le \sum_{m=1}^{\bar{m}} \tau_{m \mu_{ji}} x_{ml}, \; o_{ji} \in O^{+}, \; l \in [\![1, \bar{m}]\!] \qquad (5)$$

with

$$x_{ml} = \begin{cases} 1 \text{ if } M_m \text{ is assigned to } L_l \\ 0 \text{ otherwise} \end{cases}, \quad m \in [\![1, \bar{m}]\!], \; l \in [\![1, \bar{m}]\!],$$

$$y_{o_{ji} l} = \begin{cases} 1 \text{ if operation } o_{ji} \text{ is assigned to } L_l \\ 0 \text{ otherwise} \end{cases},$$

$o_{ji} \in O^{+}$, $l \in [\![1, \bar{m}]\!]$.

The objective function minimizes the sum of the loaded travel times (1). Constraints (2) and (3) indicate respectively that each location receives exactly one machine, and that each machine is assigned to exactly one location. Constraints (4) indicate that each operation is assigned to exactly one location. Constraints (5) indicate that one operation can be assigned to a location if and only if this location receives a machine of the appropriate type.

Peculiar case of the FMS job shop

In the case of FMS job shop scheduling problem, variables $y_{o_{ji}l}$ become useless (operations are automatically located on the suitable machine). The previous model can be simplified as indicated below:

Minimise $$\sum_{o_{ij}\in O}\sum_{l=1}^{m}\sum_{l'=1}^{m} t_{ll'} x_{\mu_{i,j-1},l} x_{\mu_{ij},l'} \quad (1')$$

Subject to $$\sum_{m=1}^{\bar{m}} x_{ml} = 1, l \in [\![1,\bar{m}]\!] \quad (2)$$

$$\sum_{l=1}^{\bar{m}} x_{ml} = 1, m \in [\![1,\bar{m}]\!] \quad (3)$$

The objective function is rewritten by using the variables x_{ml} (we recall that in the job shop context, μ_{ji} designs indifferently the machine or the type). Constraints (2) and (3) are identical to those of the first model.

This is the Koopmans & Beckman formulation for the QAP (1957), in which the coefficients of the cost matrix are all equal to 1.

Obviously, the quadratic problem formulated in the general case is NP-hard because the QAP is a special case of this problem. It can be solved exactly, if its size allows it, by mathematical solvers, or solved approximately by heuristic approaches.

The main objective of this chapter is not to optimally solve the QAP. Nevertheless, the solution of this problem is interesting for many reasons. Firstly, we want to show that the optimal solution of the QAP is not necessarily those that leads to the most efficient machine assignment. Secondly, we can expect that this optimal solution (or an approximate solution when the optimal solution is out of reach) is a good assignment. This knowledge can be useful for finding better assignments. In the remainder of this chapter, we will use the misnomer term of "*QAP-optimal assignment*" for defining the solution of the QAP, even if the problem is not optimally solved.

3.2. The assignment generation

Let us first justify the choice of a complex paradigm for generating new assignments. Starting with the QAP-optimal assignment, we try to iteratively improve it by evaluating the makespan criterion. It is necessary to generate new machine assignments. A simple way consists to modify a solution by the application of a basic move (for example the exchange of two machines) and to implement a local search technique. The drawback of this approach is that we cannot guarantee that it allows to reach the optimal machine assignment. Another simple way is to generate the assignments randomly. Obviously, each assignment is then reachable. The drawback of this approach is that we don't exploit the knowledge of the QAP optimal assignment, which is a good assignment. It is the reason why we use an ant colony system (ACS) approach for generating new solutions. Each assignment has a non null probability to be chosen, and this probability is higher for the assignments that are close to the QAP optimal assignment. We can notice that using this approach, different ants can construct the same assignment that will be evaluated several times. Far to be a problem, this means that the most promising assignments (the closest ones to the QAP optimal assignment) are evaluated several times, exactly as if we execute several runs and we keep the best evaluation.

In ACS, ants progressively build random solutions, guiding themselves with the pheromone information. We first recall how ACS is applied to the traveling salesman problem (TSP) in the literature. In a second time, we propose an original way to use the QAP-optimal assignment in order to define a

TSP-like problem on which ACS will be applied.

Ant Colony System for the Traveling Salesman Problem

Ant Colony System is a cooperative learning algorithm that was first introduced by Colorny *et al.*, 1991. Dorigo & Gambardella, (1997) describe how this approach can be applied to the traveling salesman problem (TSP).

Let *G* be a complete graph in which the vertices are the cities. The edges (r,s) are valued by a cost function $\delta(r,s)$ (which can represent a distance or a time between *r* and *s*), and by a desirability function $\tau(r,s)$ which represents the pheromone rate on this edge. The general principle of the ACS algorithm is that each ant builds a solution (an Hamiltonian cycle in the TSP context). Let's assume that an ant is in city *r*. It chooses its next city *s* among those which have not yet been visited. For that, the ant takes into account the cost of the travel and the pheromone rate between *r* and *s*.

The probability with which ant *k* in city *r* chooses to move to city *s* is given by (Eq. 1) (this state transition rule is called *random-proportional rule* in Dorigo & Gambardella, (1997)).

$$p_k(r,s)=\begin{cases}\dfrac{\tau(r,s)\eta(r,s)^{\beta}}{\sum_{u\in J_k(r)}\tau(r,u)\eta(r,u)^{\beta}} & \text{if } s\in J_k(r)\\ 0 \text{ otherwise}\end{cases}$$

(Eq. 1)

where τ is the pheromone, $\eta=1/\delta$ is the inverse of the distance $\delta(r,s)$, $J_k(r)$ is the subset of cities that remain to be visited by ant *k* located in city *r*, β is a parameter which determines the relative importance between pheromone and distance.

Each time an ant ends the construction of its tour, a local updating rule (Eq. 2) is applied to change the pheromone level.

$$\tau(r,s)\leftarrow(1-\rho)\tau(r,s)+\rho\tau_0 \quad \text{(Eq. 2)}$$

where $0<\rho<1$, and τ_0 is the initial level of pheromone.

An iteration of ACS ends when all ants have constructed their solution. A global updating rule (Eq. 3) is then applied using the globally best solution found since the beginning of the trial.

$$\tau(r,s)\leftarrow(1-\alpha)\tau(r,s)+\alpha\Delta\tau(r,s) \text{ (Eq. 3)}$$

where

$$\Delta\tau(r,s)=\begin{cases}\left(L_{gb}\right)^{-1} \text{ if } (r,s)\in \textit{gbest\_solution}\\ 0 \text{ otherwise}\end{cases}$$

$0<\alpha<1$ is a parameter and L_{gb} is the length of the best solution encountered so far.

Dorigo & Gambardella (1997) propose also the following parameter settings: $\beta=2$, $\alpha=\rho=0.1$, $\tau_0=(nL_{nn})^{-1}$ where *n* is the number of cities and L_{nn} is a rough approximation of the optimal value (for example obtained by a construction heuristic). The number of ants is $m=10$. The initial level of pheromone is given by $\tau(r,s)=\tau_0$ for any pair (r,s) of cities.

The last remark concerns the state transition rule called *pseudo-random-proportional rule* (Eq. 1') in which q_0 is a parameter ($0\le q_0\le 1$). q_0 determines the relative importance of exploitation (high values of q_0) versus exploration (low values of q_0). The *random-proportional rule* is obtained by fixing $q_0=0$ in Eq. 1'. In this chapter, we will apply the *random-proportional rule* because we will consider very small instances of TSP (between 5 and 13 cities).

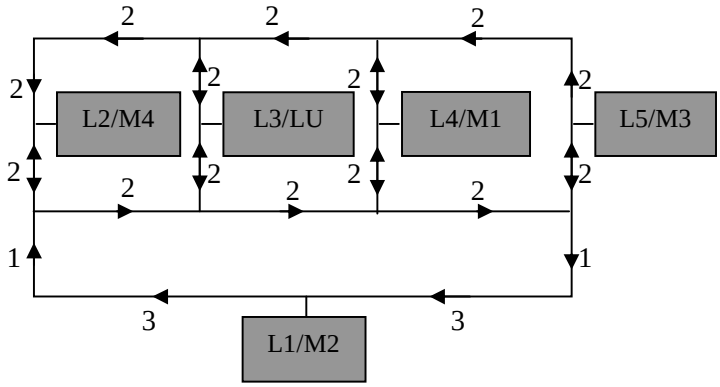

Fig. 3. *AP-optimal assignment for the Ex11 instance (section 4.1).*

$$p'_k(r,s)=\begin{cases}\arg\max\limits_{u\in J_k(r)}\left\{\tau(r,u)\eta(r,u)^{\beta}\right\} & \text{if } q\le q_0\\ p_k(\text{r,s}) \text{ otherwise}\end{cases} \qquad \text{(Eq. 1')}$$

Adaptation of ACS for the machine assignment problem

By applying the scheme just defined above, ants must build a machine assignment. The driving principle we support here is to employ the QAP-optimal assignment to define a TSP. For that, we consider a bipartite graph (L,M,E) where the vertices are defined by the set of locations *L* on the left side and by the set of machines *M* on the right side. Each edge (l,m) for $L_l \in L$ and $M_m \in M$ is valued by a distance $\delta(l,m)$ defined below. Each ant randomly chooses successively the machine that is assigned to the location L_1, the machine that is assigned to the location L_2, …, the machine that is assigned to the location $L_{\bar{m}}$. We illustrate the different steps of this problem transformation on the instance Ex11[2]. The reader can refer to the table 4. The QAP-optimal solution for this instance is $\sigma(1,2,3,4,5)=(2,4,5,1,3)$, where $\sigma(l)=m$ indicates that the machine M_m is located at L_l (5 designs here the L/U station). The QAP-optimal solution is represented in figure 3. We propose to define a distance between a location and a machine by the formula $\delta(l,m)=t_{ll'}$, for $l\in[\![1,\bar{m}]\!]$ and $m\in[\![1,\bar{m}]\!]$, where $\sigma(l')=m$. Unfortunately, if we consider this definition, we have obviously $\delta(l,\sigma(l))=0$. By applying this result at (Eq. 1), we obtain $p_k(l,\sigma(l))=1$, and ants will build invariably the same machine assignment. In order to avoid this drawback, we propose to take $\delta(l,\sigma(l))=\gamma\min\limits_{m\neq\sigma(l)}(\delta(l,m))$, where $0<\gamma\le 1$ is a parameter. In fact, the setting of value for γ creates a balance between intensification and diversification. If γ is close to 0 then the probability to assign the machine $M_{\sigma(l)}$ to the location L_l is high. In the opposite case, if γ is close to 1, this probability is low. The special case $\gamma=1$ means that we have equiprobability to assign $M_{\sigma(l)}$ to the location L_l and to the closest location of L_l.

For the considered example, the makespan of the QAP-optimal solution has been evaluated at 98. So, we can calculate $\tau_0=(5*98)^{-1}$, $\delta(1,3)=t_{15}=12$, $\delta(1,4)=t_{12}=6$, $\delta(1,5)=t_{13}=8$ and $\delta(1,2)=\gamma\times 6$.

Table 2. *Probability* $p_k(l,m)$ *to assign machine* M_m *to location* L_l *according to* γ.

	$p_k(1,1)$	$p_k(1,2)$	$p_k(1,3)$	$p_k(1,4)$	$p_k(1,5)$
$\gamma=0.2$	0.013	0.920	0.009	0.037	0.021
$\gamma=0.5$	0.058	0.648	0.041	0.162	0.091
$\gamma=0.8$	0.096	0.418	0.067	0.268	0.151
$\gamma=1.0$	0.113	0.315	0.079	0.315	0.177

Table 2 summarizes the obtained probabilities by applying (Eq. 1) with $\beta=2$ for different settings of γ.

In order to define the best setting for γ, a preliminary experiment has been realised on the instance Ex21. This instance has been chosen because the QAP-optimal assignment is not of very good quality. We apply ACS in the same conditions, except for the setting of γ. For each value of γ, five runs have been executed. The table 3 gives the average results with regards to the relative gap to the best known upper bound.

We can see that higher the value of γ is, better the performance of ACS is. We will use in the following $\gamma=1$. Although we may obtain better results with $\gamma>1$, we prevent this case because it is illogic with the definition of $\delta(1,m)$. This experiment seems to show that it is preferable to favour diversification for this parameter.

Table 3. *Relative performance of ACS with regards to the setting of* γ.

γ	0.1	0.2	0.3	0.4	0.5
gap	9.43%	9.25%	9.25%	6.42%	4.53%
γ	0.6	0.7	0.8	0.9	1.0
gap	4.53%	3.77%	3.77%	3.40%	1.13%

3.3. The Assignment evaluation

For evaluating a machine assignment, we consider now the simultaneous scheduling of machines and vehicles in order to minimize the makespan criterion.

In the case of the FMS job shop problem, the related problem has been formulated as a nonlinear mixed integer programming model by Bilge & Ulusoy (1995). It is denoted $JR|t_{kl},t'_{kl}|C_{\max}$ following the notation of Graham *et al*. (1979), extended by Knust (1999) for transportation problems. *J* indicates a job shop, *R* indicates that we have a limited number of identical vehicles and all jobs can be transported by any of the vehicles. t_{kl} indicates that we have job-independent, but machine-dependent loaded travel times. t'_{kl} indicates that we have machine-dependent empty travel times. The objective function to minimize is the makespan C_{max}. This problem is related to the Vehicle Scheduling Problem (VSP) for the scheduling of vehicles, and to the Job Shop Problem (JSP) for the scheduling of machines. The difficulty of $JR|t_{kl},t'_{kl}|C_{\max}$ is the joint consideration of these two NP-hard problems.

(Ulusoy *et al*., 1997; Abdelmaguid *et al*., 2004) propose genetic algorithms to tackle this problem. (Deroussi *et al*., 2008) develop an iterative local search metaheuristic. The authors propose to schedule transportation tasks on vehicles. A schedule is evaluated using a simple discrete event approach (we recall that $JR|t_{kl},t'_{kl}|C_{\max}$ problem is offline; traffic congestions or collisions are ignored). We will call DGT this metaheuristic in the following of this chapter. It will be used as a blackbox subroutine for the assignment evaluations.

We introduce now the main principles of DGT. Given a machine assignment and a list of jobs to produce, then the list of transportation tasks is known. For example, if a job is processed by the following sequence of machines M_3,M_1,M_2, we will define four transportation tasks for this job, from the load station to M_3, from M_3 to M_1, from M_1 to M_2, and finally from M_2 to the unload station. A solution in DGT is

represented by the sequence of transportation tasks that is assigned to each vehicle. The objective of DGT is to determine the best possible scheduling of transportation tasks. For that, a neighborhood is defined for modifying the current solution. An evaluation subroutine based on discrete events calculates the arrival dates of jobs on the machine input buffer, and the completion time of each operation. The makespan is equal to the completion time of the last operation.

An important advantage can be pointed out by using this separation between optimization and evaluation. Some dispatching rules can be added or removed depending on the evaluation model requirements. In our case, the studied problem is typically at strategic level. This means that we will have to work with aggregated data. In this condition, considering online scheduling rules such as traffic constraints probably makes no practical sense. So, we can use a relatively simple evaluation model. Moreover, it has been fairly easy to modify our model for evaluating solutions for the FMS flexible job shop. When a vehicle starts a transportation task, the destination machine must be determined according to the type required by the job. The following rule is applied; the chosen machine is the one that processes the job in the shortest time. Except for this difference, the evaluation models are exactly the same for the two FMS problems.

DGT is an iterated local search algorithm. Unless otherwise stated, 1000 local searches are executed for each evaluation of a machine assignment.

4. COMPUTATIONAL STUDY FOR THE FMS JOB SHOP PROBLEM

We propose now to implement the proposed solution approach in order to determine machine assignments for the FMS job-shop problem. The benchmark test is composed by instances derived from the literature, and described in the first subsection. The main objectives of this part are to show that, for these toy instances (there are only 120 possible machine assignments), the solution of the QAP is not always the most effective, and that ACS is appropriate for searching better solutions. We will give in the second subsection the QAP-optimal assignments. These assignments are optimal for the minimisation of the loaded vehicle travel distance. The third subsection is devoted to the exhaustive enumeration of assignments. We will show that the QAP-optimal solution is not the most efficient in many cases. We finish in the forth subsection by a comparison between two methods that generate new machine assignments: ACS and local search.

All the experiments in sections 4 and 5 are executed on a Pentium 4, 3.4 GHz. Routines are written in C.

4.1. Benchmark Test

We will adapt the benchmark test initially proposed by Bilge & Ulusoy (1995). It is composed of 40 instances, built from 4 layouts (including the layout 1 shown in figure 1) and 10 jobsets (jobset 1 is illustrated by figure 4). Each layout contains 5 machines (4 production machines noted M_1, M_2, M_3, M_4 and a single load / unload station noted LU) and two vehicles (AGVs). Jobsets contain between 5 and 8 jobs, and between 18 and 27 operations to schedule. The name "Ex_*ij*" will design the instance built from jobset *i* and layout *j*.

Loaded and empty vehicle travels are supposed to have the same durations (in particular, loaded and unloaded times are neglected). So, layout data is simply composed of the matrix of the vehicle travel times ($t'_{ll'} = t_{ll'}$).

The jobsets give, for each job, the sequence of operations that have to be processed in the given order ($\mu_{ji} \in M$, $p_{ji} \in \mathbb{N}$). Compared to the initial jobsets, we add a fictitious operation at the end of the process plan of each job. This additional operation takes place on the unload station, and have null duration $(LU,0)$. It ensures that each job physically leaves the system after being processed.

A solution is given by a *m*-tuple where the i^{th} component designs the machine which is assigned to the location L_i. For example, the assignment (M_1,M_4,LU,M_3,M_2), or $(1,4,LU,3,2)$ indicates that M_1 is assigned to L_1, M_4 to L_2, and so on. We assume that there are no incompatibility issues between the machines and the locations.

Each of these instances contains 5 machines to assign to 5 locations. Thus, we have 5!=120 possible machine assignments. The QAP-optimal assignment can be easily obtained by exhaustive enumeration.

$$\left\{\begin{array}{l}\{(M_1,8),(M_2,16),(M_4,12),(LU,0)\}\\ \{(M_1,20),(M_3,10),(M_2,18),(LU,0)\}\\ \{(M_3,12),(M_4,8),(M_1,15),(LU,0)\}\\ \{(M_4,12),(M_2,18),(LU,0)\}\\ \{(M_3,10),(M_1,15),(LU,0)\}\end{array}\right\}$$

Fig. 4. *Sequence of operations to be processed in the given order in jobset 1.*

4.2. The QAP machine layout

In table 4, we indicate in column "QAP Assign" for each instance, the optimal assignment according to the minimization of the loaded vehicle travel distances. The column "Cost" gives the value of this criterion. These QAP-optimal solutions give a first machine assignment, which is not necessarily optimal when empty vehicle travels are also considered.

Table 4. *The QAP-optimal assignments and the loaded vehicle travel distance.*

Instances	QAP Assign	Cost	Instances	QAP Assign	Cost
Ex11	(2,4,LU,1,3)	182	**Ex61**	(LU,1,2,3,4)	156
Ex12	(1,3,4,2,LU)	114	**Ex62**	(LU,1,2,3,4)	60
Ex13	(4,LU,1,3,2)	140	**Ex63**	(LU,1,2,3,4)	84
Ex14	(3,2,4,1,LU)	214	**Ex64**	(2,3,4,1,LU)	168
Ex21	(LU,1,2,3,4)	144	**Ex71**	(2,3,4,LU,1)	186
Ex22	(1,2,3,4,LU)	60	**Ex72**	(LU,1,2,3,4)	84
Ex23	(LU,1,2,3,4)	84	**Ex73**	(LU,1,2,3,4)	112
Ex24	(3,4,LU,2,1)	156	**Ex74**	(3,4,LU,2,1)	204
Ex31	(4,LU,1,2,3)	152	**Ex81**	(LU,1,2,3,4)	164
Ex32	(1,2,3,4,LU)	66	**Ex82**	(LU,1,2,3,4)	60
Ex33	(LU,1,2,3,4)	98	**Ex83**	(LU,1,2,3,4)	84
Ex34	(1,2,3,LU,4)	166	**Ex84**	(1,2,3,LU,4)	160
Ex41	(1,3,2,LU,4)	172	**Ex91**	(1,2,4,LU,3)	146
Ex42	(3,LU,4,1,2)	96	**Ex92**	(4,LU,3,1,2)	60
Ex43	(LU,1,2,3,4)	140	**Ex93**	(1,2,4,LU,3)	84
Ex44	(4,1,3,2,LU)	200	**Ex94**	(1,2,4,3,LU)	148
Ex51	(3,4,2,1,LU)	130	**Ex101**	(LU,1,2,3,4)	184
Ex52	(3,4,1,2,LU)	74	**Ex102**	(LU,2,1,3,4)	84
Ex53	(4,1,2,LU,3)	98	**Ex103**	(2,1,3,4,LU)	112
Ex54	(3,2,4,LU,1)	152	**Ex104**	(1,3,4,2,LU)	190

Table 5. *Comparison of best, QAP-optimal and worst assignments for the makespan criterion.*

Instances	Best Assign	C_{best}	C_{QAP}	C_{worse}	Instances	Best Assign	C_{best}	C_{QAP}	C_{worse}
Ex11	(2,4,LU,1,3)	98	98	118	**Ex61**	(3,4,LU,2,1)	123	129	154
Ex12	(LU,1,3,4,2)	82	84	101	**Ex62**	(LU,1,2,3,4)	102	102	142
Ex13	(LU,3,4,1,2)	91	92	104	**Ex63**	(LU,1,2,3,4)	105	105	154
Ex14	(4,1,2,LU,3)	114	119	150	**Ex64**	(1,2,3,LU,4)	132	135	199
Ex21	(3,4,LU,2,1)	106	116	136	**Ex71**	(2,3,4,LU,1)	124	124	164
Ex22	(LU,1,2,3,4)	82	86	124	**Ex72**	(LU,1,2,3,4)	86	86	149
Ex23	(LU,1,2,3,4)	89	89	131	**Ex73**	(LU,1,2,3,4)	93	93	160
Ex24	(1,2,3,LU,4)	118	120	176	**Ex74**	(1,2,3,LU,4)	138	141	218
Ex31	(4,1,LU,2,3)	114	117	138	**Ex81**	(LU,1,2,3,4)	167	167	179
Ex32	(1,2,3,4,LU)	89	89	125	**Ex82**	(LU,1,2,3,4)	155	155	179
Ex33	(LU,1,2,3,4)	96	96	133	**Ex83**	(LU,1,2,3,4)	155	155	183
Ex34	(1,2,3,LU,4)	121	121	177	**Ex84**	(1,2,3,LU,4)	165	165	228
Ex41	(1,3,2,LU,4)	123	123	145	**Ex91**	(1,2,4,LU,3)	115	115	147
Ex42	(3,LU,4,1,2)	94	94	120	**Ex92**	(LU,3,1,2,4)	97	99	132
Ex43	(LU,1,2,3,4)	102	102	128	**Ex93**	(1,2,4,LU,3)	101	101	142
Ex44	(1,2,3,LU,4)	140	142	185	**Ex94**	(1,4,3,LU,2)	125	126	181
Ex51	(3,2,4,LU,1)	94	97	113	**Ex101**	(4,LU,2,1,3)	148	153	171
Ex52	(2,LU,3,4,1)	73	74	96	**Ex102**	(LU,2,1,3,4)	132	132	158
Ex53	(4,1,2,LU,3)	81	81	101	**Ex103**	(2,1,3,4,LU)	135	135	168
Ex54	(3,2,4,LU,1)	107	107	145	**Ex104**	(2,1,3,LU,4)	158	159	214

4.3. Exhaustive enumeration of machine assignments

As there are only 120 possible machine assignments, we propose to evaluate each of them according to the makespan criterion (using the black-box subroutine DGT). For each assignment, 10 runs are realised (this represents 1200 calls of DGT for each instance). Each instance requires between 40 and 100 minutes of CPU time (one call of the makespan evaluation requires between 2 and 5 seconds according to the size of the instance). We present in table 5 the results obtained. The column "**Best Assign**" gives the best obtained machine assignment. Its cost is given in column "$\mathbf{C_{best}}$". The columns "$\mathbf{C_{QAP}}$" and "$\mathbf{C_{worse}}$" gives respectively the best result obtained with QAP-optimal assignment and with the worst assignment (the latter is given for information only).

We can notice that the QAP-optimal machine assignment is not the best one for 17 of the 40 instances that compose our benchmark test (these instances are indicated in the table with grey-colored cells). This means that for about half of the instances, the QAP-optimal machine assignment can be improved. However, the QAP-optimal assignment is generally of good quality. This justifies the choice of QAP assignment as initial solution. To conclude, we can notice the case of the instance Ex21, for which the QAP-optimal solution has a makespan of 116 against 106 for the best assignment (this represents a relative gap of 8.62%). We give in figure 5 the Gantt charts of these two solutions. Of course, the sum of loaded vehicle travel distance is better for the QAP-optimal solutions (144 against 150). On the contrary, the QAP-optimal solution is worse when we compare the sum of vehicle travel distance (224 against 208).

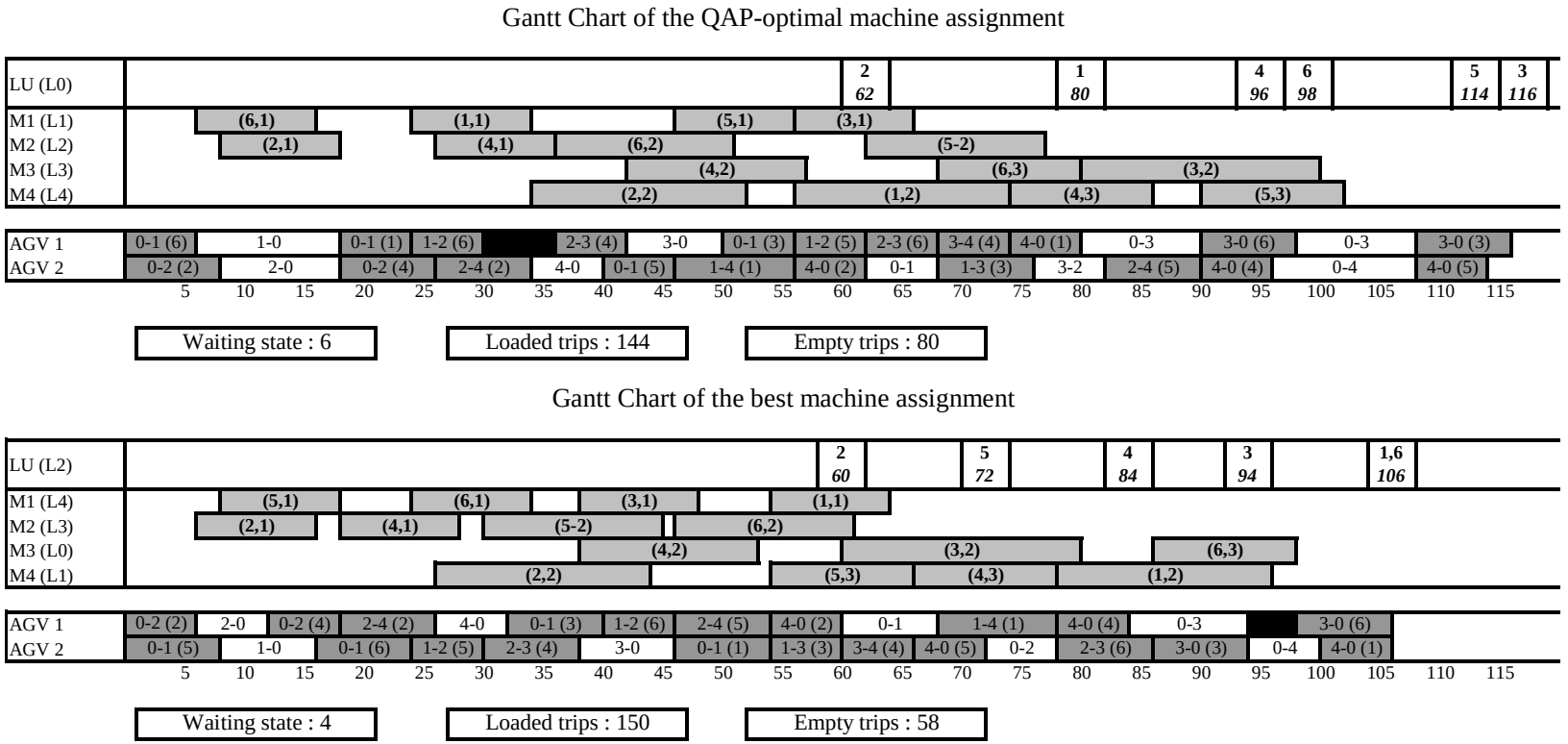

Fig. 5. *Gantt charts for the QAP-optimal and the best assignments on the instance Ex21.*

4.4. Comparison ACS / optimisation method

We focus in this section on the 17 instances for which the QAP-optimal assignment can be improved. In a preliminary study, we firstly apply classical optimization methods such as local search (figure 6). 10 runs have been executed for each instance in order to prevent stochastic variations. The heuristic stops after 120 calls at the evaluation subroutine DGT (we have a total of 1200 calls of DGT, the same as for the enumeration). The perturbation of a machine assignment is a random exchange move. Let for example X=(1,3,2,5,4) be a machine assignment. If we randomly choose to exchange machines 5 and 3, we obtain the machine assignment Y=(1,5,2,3,4). We can remark that we allow to exchange the same machine. This leads to Y=X. In this case, we re-evaluate the current best found assignment. It is necessary because several evaluations of the same assignment can give different makespans.

```
X ← X_QAP
Cx ← DGT(X)
While stopping conditions are not met Do
    Y ← Perturbation(X)
    Cy ← DGT(Y)
    If (Cy<=Cx) Then
        X ← Y
        Cx ← Cy
    End If
End While
```

Fig. 6. *Pseudo-algorithm of the implemented local search.*

We summarize in table 6 the obtained results. The columns "$\mathbf{C_{best}}$" and "$\mathbf{C_{QAP}}$" indicate the best makespan obtained respectively for the best and for the QAP-optimal assignments. For the local search, we have three columns "**Min**", "**Max**" and "**Avg**", in which we give respectively the result of the best run, of the worst run and the average result of the 10 runs. The row "**Gap**" indicates the relative gap of the method to the best found evaluation.

The same information is given for the ACS algorithm. 100 iterations are performed (with 10 ants). Each run of ACS requires 1000 calls of the evaluation subroutine DGT (against 120 calls for local search). In order to give the same time for the two methods, we divide the time given for one evaluation of a machine evaluation by a factor 8. So, the makespan evaluation for a machine assignment is of less good quality. We give in the column "**Time**" the approximate computational time to execute one run of LS and ACS (in seconds).

We can remark that ACS performs better than LS (ACS obtains a global performance of 0.78% against 1,81% for LS). LS improves the QAP-optimal solution for only 6 of the 17 instances (grey colored cells), and the best assignment is found for 5 instances (dark grey colored cells). These results are relatively disappointing. Two reasons can be put forward:

- the good quality of best QAP assignment. For example, for Ex24, there is only one better assignment. This explains the difficulty to improve the initial solution,
- the exchange move doesn't seem efficient. For Ex13, there are 5 better assignments, but none of them can be reached from the QAP-optimal assignment by an exchange move.

This last point is particularly worrying. This seems to indicate that LS is dependent on the landscape of the instance: if it exists a better solution in the neighborhood, LS will be effective; if not, LS won't be able to improve the QAP-optimal solution. It seems difficult to define a criterion that allows to foresee the landscape of each instance.

Table 6. *Comparison of local search and Ant colony system.*

	C_{best}	C_{QAP}	LS			ACS			Time (s)
Instances			Min	Max	Avg	Min	Max	Avg	
Ex12	82	84	82	82	82.0	82	84	82.2	250
Ex13	91	92	92	92	92.0	91	92	91.1	250
Ex14	114	119	114	114	114.0	114	115	114.3	250
Ex21	106	116	106	116	110.2	106	106	106.0	430
Ex22	82	86	86	86	86.0	86	86	86.0	430
Ex24	118	120	120	120	120.0	120	120	120.0	430
Ex31	114	117	114	114	114.0	114	116	114.4	420
Ex44	140	142	142	142	142.0	140	140	140.0	420
Ex51	94	97	94	97	94.3	94	94	94.0	250
Ex52	73	74	74	74	74.0	73	74	73.8	250
Ex61	123	129	129	129	129.0	123	125	124.0	500
Ex64	132	135	135	135	135.0	133	134	133.1	500
Ex74	138	141	141	142	141.8	136	141	137.9	680
Ex92	97	99	99	99	99.0	97	97	97.0	330
Ex94	125	126	126	126	126.0	126	127	126.1	330
Ex101	148	153	152	152	152.0	149	151	150.2	570
Ex104	158	159	159	159	159.0	159	159	159.0	570
gap		2.87%			1.81%			0.78%	

On the contrary, the results obtained with ACS are promising. In most cases, ACS permits to improve the QAP-optimal solution (for 13 of the 17 instances). The best known assignment is found for 12 instances. The most interesting point is that ACS appears to be a robust method. The standard deviation is very low. We can note that a best upper bound has been found for the Ex74 instance.

In the next section, we will study in more detail the behaviour of ACS for more difficult instances, for which it is impossible to enumerate the machine assignments.

5. COMPUTATIONAL STUDY FOR FMS FLEXIBLE JOB SHOP PROBLEM

We propose to consider now the FMS flexible job shop problem. We describe firstly the benchmark test, before we give the obtained *QAP-optimal* solutions. The second subsection is dedicated to the calculation of the makespan of these *QAP-optimal* solutions. The results obtained by applying ACS are finally presented before we conclude regarding the experimental part of this work.

5.1. Benchmark test

We propose a new family of instances inspired from the literature. It derives from the Bilge & Ulusoy's instances for the FMS job shop problem. These instances have been adapted in the following way: it is assumed that each production machine is duplicated. So we have two identical machines of each type, except for the LU station. The FMS is composed of 9 locations. We have modified the layout 1 by adding 4 locations, as it is shown in figure 7. The transportation time matrix is deduced from this modified layout by determining

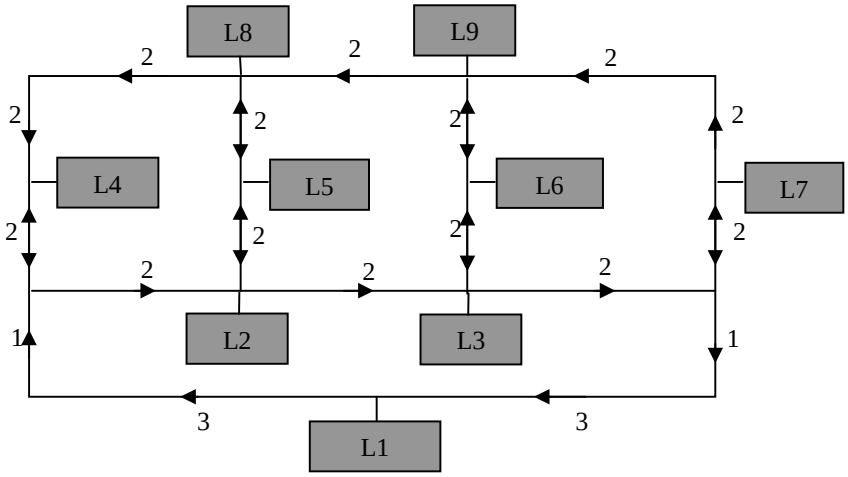

Fig. 7. *The 9-locations machine layout, used for FJSP-BU instances.*

the shortest path between two locations. In order to restore the balance between transportation times and production times, the latter are increased twofold in every jobset. We will denote these instances by FJSP-BU'i', where 'i' designs the jobset. All these instances are solved with 2 vehicles.

5.2. Solution of the QAP

All of these instances have been solved with the mathematical solver CPLEX using the MIP formulation described in section 3.1. For each instance, CPLEX has been stopped after 3 hours of computational time. None of the ten instances has been optimally solved. Nevertheless, the solver gives the best found machine assignment when the time is up. The obtained results are presented in table 7. Despite the fact that the given solutions are not proved to be optimal, we will continue to call these solutions the "*QAP optimal*" assignments in order to keep homogeneous designation in the chapter. These solutions are good enough, in order to be used as initial assignment.

Table 7. *The "QAP-optimal" assignments and the loaded vehicle travel distance.*

Instances	QAP Assign	Cost
FJSP-BU11	(1,3,8,4,6,LU,5,7,2)	88
FJSP-BU21	(1,2,4,6,LU,3,7,8,5)	68
FJSP-BU31	(4,3,5,6,1,8,7,LU,2)	68
FJSP-BU41	(1,LU,4,7,6,8,5,3,2)	72
FJSP-BU51	(2,1,6,8,LU,4,3,5,7)	60
FJSP-BU61	(2,7,LU,5,6,4,8,3,1)	72
FJSP-BU71	(6,LU,1,7,8,4,2,3,5)	88
FJSP-BU81	(1,8,LU,6,5,4,7,3,2)	72
FJSP-BU91	(6,5,1,8,3,4,2,LU,7)	60
FJSP-BU101	(7,LU,4,2,8,1,6,3,5)	80

5.3. Evaluation of the QAP-optimal solutions

We use the evaluation subroutine DGT to evaluate the "*QAP-optimal*" assignments. For each instance, 10 runs are completed. We present in table 8 the obtained results. The columns "**Min**", "**Max**" and "**Avg**" give respectively the best, the worst and the average makespan. The time in seconds is indicated in the column "**Time**". The row "**Gap**" gives the average relative gap to the best known solution.

Table 8. *Evaluation of the "QAP optimal" assignment for the makespan criterion*

Instances	Min	Max	Avg	Time
FJSP-BU11	138	138	138.0	50
FJSP-BU21	112	112	112.0	30
FJSP-BU31	126	128	126.4	35
FJSP-BU41	114	116	114.2	40
FJSP-BU51	94	96	95.2	20
FJSP-BU61	140	142	140.8	45
FJSP-BU71	108	112	110.0	70
FJSP-BU81	182	182	182.0	50
FJSP-BU91	140	142	140.6	30
FJSP-BU101	174	178	175.2	50
			4.36%	

5.4. Application of ACS

We apply ACS on the FMS-flexible job shop instances. The results obtained are summarized in table 9, which is similar to the table 8. The grey-colored cells indicate that ACS has found a better assignment than the *QAP-optimal* assignment

Table 9. *Optimization by ACS.*

Instances	Min	Max	Avg	Time
FJSP-BU11	134	136	134.8	400
FJSP-BU21	110	112	110.8	210
FJSP-BU31	116	118	117.2	230
FJSP-BU41	108	112	110.0	260
FJSP-BU51	90	90	90.0	110
FJSP-BU61	136	138	136.8	300
FJSP-BU71	106	112	109.2	470
FJSP-BU81	174	175	174.8	350
FJSP-BU91	136	138	136.4	210
FJSP-BU101	170	174	171.2	400
Gap			0.93%	

ACS finds always a better assignment. Sometimes, the improvement is significant (116 against 126 for the BU31 instance). The average results show the efficiency of ACS (ACS is at 0.93% of the best found solution against 4.36% for the QAP optimal solution. It could be expected a significant long term benefits in term of productivity, if our general approach is applied for this kind of FMS.

6. CONCLUSION

We focus in this study on the machine assignment problem in flexible manufacturing systems environment. We show that the simple fact to consider empty travel times leads in many cases to the design of better layouts. Unfortunately, empty travel times are very difficult to determine because these times require a dynamic evaluation. This leads either to consider very simple systems, or to propose an elaborated solution approach. We propose in this paper a hybrid approach that combines integer linear programming, metaheuristics and discrete events approaches.

Actually, the link between design problems and scheduling problems is not sufficiently studied in the FMS literature. It constitutes an important research topic, with possible industrial spin-offs.

7. REFERENCES

Abdelmaguid, T.F., Nassef, O.N., Kamal, B.A., & Hassan, M.F. (2004). A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 42, 267-281.

Asef-Vaziri, A., Laporte, G., & Ortiz, R. (2007). Exact and Heuristic procedures for the material handling circular flow path design problem. *European Journal of Operational Research* 176, 707-726.

Asef-Vaziri, A., Hall, N. G., & George, R. (2008). The significance of deterministic empty vehicle trips in the design of a

unidirectional loop flow path. *Computers & Operations Research*, 35, 1546-1561.

Barnes, J. W., & Chambers, J. B. (1996). *Flexible Job Shop Scheduling by Tabu Search* (Tech Rep. Series, ORP96-09). Austin, USA: University of Texas, Graduate Program in Operations Research and Industrial Engineering.

Bhattacharya, A., Abraham, A., & Vasant, P. (2008). FMS Selection Under Disparate Level-of-Satisfaction of Decision Making Using an Intelligent Fuzzy-MCDM Model. In C. Kahraman (ed.), *Fuzzy Multi-Criteria Decision Making*, (pp. 263-280).

Bilge, U, & Ulusoy, G. (1995). A time window approach to simultaneous scheduling of machines and material handling system in an FMS. *Operations Research*, 43, 1058-1070.

Colorny, A., Dorigo, M., & Maniezzo, V. (1991). Distributed Optimization by Ant Colony. *European Conference on Artificial Life*. Paris, France.

Deroussi; L., Gourgand, M. & Tchernev, N. (2008), A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 46, 2143-2164

Dorigo, M., & Gambardella, L. M., (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1, 53–66.

Drira, A., Pierreval, H., & Hajri-Gabouj, S., (2006). Facility layout problems: A survey. *12th IFAC Symposium on Information Control Problems in Manufacturing*.

El-Baz, M.A., (2004). A Genetic Algorithm for Facility Layout Problems of Different Manufacturing Environments. *Computers & Industrial Engineering*, 47, 233-246.

Ganesharajah, T., Hall, N. G. & Sriskandarajah., C. (1998). Design and operational issues in AGV-served manufacturing systems. *Annals of Operations Research*, 76, 109-154.

Graham, R.L., Lawler, E.L., Lenstra, J.K., & Rinnooy Kan, A.H.G., (1979). Optimization and approximation in deterministic sequencing and scheduling, *Annals of Discrete Mathematics*, 5, 287–326.

Kim, J., & Goetschalckx, M., (2005). An integrated approach for the concurrent determination of the block layout and I/O point locations based on the contour distance. *International Journal of Production Research*, 43, 2027-2047.

Koopmans, T.C., & Beckman, M., (1957). Assignment Problems and the Location of Economic Activities. *Econometric*, 25, 53-76.

Knust, S., (1999). *Shop scheduling problems with transportation*. PhD thesis, Fachbereich Mathematik/ Informatik Universität Osnabrück, Germany.

Le-Anh, T., & De Koster, M.B.M., (2006). A review of design and control of automated guided vehicle systems, *European Journal of Operational Research*, 171, 1-23.

Ray, T., & Sarker, R., (2008). EA for solving combined machine layout and job assignment problems, *Journal of Industrial and Management Optimization*, 4, 631-646.

Sun, X.-C., & Tchernev, N., (1996). Impact of empty vehicle flow on optimal flow path design for unidirectional AGV systems. *International Journal of Production Research*, 34, 2827-2852.

Ülusoy, G., Sivrikaya-Serifoglu, F., & Bilge, U., (1997). A genetic algorithm approach to the simultaneous scheduling of stations and automated guided vehicles, *Computers Operations Research*, 24, 335-35