A TABU SEARCH ALGORITHM FOR SOLVING A BICRITERIA FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Geoffrey Vilcot [*,**] **Jean-Charles Billaut** [*]
Carl Esswein [*]

[*] *Université François-Rabelais de Tours*
Laboratoire d'Informatique, 64 avenue Jean Portalis,
37200 Tours, France
[**] *Volume Software*
20 rue Dublineau, BP 2630 - 37026 Tours cedex 1, France

Abstract: We consider in this paper a scheduling problem issued from an industrial context. This problem can be seen as a flexible job shop scheduling problem. Solutions are evaluated using two criteria: the makespan and the maximum lateness. We propose a Tabu search algorithm that finds a solution with minimum makespan, respecting a given bound for the maximum lateness. This approach is called the epsilon-constraint approach in multicriteria literature and it can be used for finding the whole set of dominant criteria vectors. The Tabu search algorithm is tested on benchmark instances of the literature and results are discussed.

Keywords: scheduling, flexible job shop, bicriteria, Tabu search

1. CONTEXT

In printing or boarding industry, resources can be used for processing different types of operations and operations of jobs can be performed by several resources. Routings of jobs are not identical. Scheduling the jobs can be done by solving a flexible job shop scheduling problem. Furthermore, for some practical reasons, in this kind of industry decision makers are not only interested in a single objective function, but by solutions of best compromise between criteria like makespan or flowtime and criteria involving tardiness measures.

We can found in (Jurisch, 1992) a branch-and-bound algorithm and some heuristic algorithms for the multi-purpose machine job shop scheduling problem, that is a similar problem. In (Dauzère-Pérès and Paulli, 1997) the authors propose a Tabu search for the flexible job shop problem which the makespan as objective function. A hierarchical approach can be found in (Paulli, 1995), where the first step consists in solving the assignment problem while the second step solves the job shop scheduling problem. In (Alvarez-Valdes *et al.*, 2005) the authors propose a heuristic algorithm to solve a flexible job shop. Their problem is dedicated to a glass factory industry with special constraints like *no-wait* or *overlapping*. Their objective function is to find a schedule with earliness and tardiness considerations. The authors use a two step algorithm, in the first step priority rules are used to solve the problem, in the second step a local search improves this solution. We can found in (Mastrolilli and Gambardella, 2000) two neighborhood structures for the flexible job shop problem. One interest in the neighborhood they define is that any feasible solution having an empty neighborhood is optimal.

We consider that the makespan and the maximum lateness have to be minimized. To the best of our knowledge, the bicriterion flexible job shop scheduling problem that we consider as never been addressed in the literature before (T'kindt and Billaut, 2002). More precisely, we search for a solution respecting a given bound for the maximum lateness and with minimum makespan, which is called the ϵ-constraint approach. The algorithm we propose to solve this problem is a Tabu search which is, as far as we know, an original method for such an approach.

In Section 2 we define the problem more formally and give the notations. In Section 3 we describe the Tabu search algorithm. Computational experiments are presented and discussed in Section 4.

2. PROBLEM DEFINITION AND NOTATIONS

We consider a Flexible Job shop Scheduling (FJS) problem, which is a generalization of the classical job shop problem. A set $\mathcal{J}$ of n jobs have to be scheduled on a set $\mathcal{M}$ of m disjunctive resources. Each job J_j needs n_j consecutive operations for being completed and operation i of job J_j is denoted by $O_{i,j}$. Each operation can be processed by more than one resource. We denote by $\mathcal{R}_{i,j}$ the set of resources that can perform $O_{i,j}$. We assume that the processing time of $O_{i,j}$ does not depend on the performing resource. This processing time is denoted by $p_{i,j}$. To each job J_j is associated a due date denoted by d_j.

The problem is to fix a performing resource and a starting time for each operation. We denote by C_j the completion time of J_j. To measure the quality of a schedule, two criteria are considered: the makespan $C_{max} = \max_{J_j \in \mathcal{J}} C_j$ and the maximum lateness $L_{max} = \max_{J_j \in \mathcal{J}} (C_j - d_j)$.

The problem under consideration is clearly strongly NP-hard.

Several methods can be used to find the set of non dominated solutions. We propose to use the ϵ-constraint approach: one criterion is minimized whereas the other one is bounded by ϵ. A set of solutions that represent non-dominated criteria vectors can be obtained by modifying iteratively the bound ϵ. This paper only focus on the algorithm for solving the ϵ-constraint formulation of the problem, that is that we search for a solution that minimizes criterion C_{max} and that respects the constraint $L_{max} \leq \epsilon$.

3. TABU SEARCH ALGORITHM

Tabu search algorithms require an *initial solution* and a *neighborhood structure* and proceed by transiting from one solution to another using *moves*. All the neighbors of a current solution are examined and the best *admissible move* is selected. Note that this move may decrease the quality of the solution. An admissible move is not in the list that contains forbidden moves, so called the *tabu list*. A general description of Tabu search algorithms can be found in (Glover, 1989).

3.1 Initial solution

The initial solution is found by using a two-step procedure: firstly, an assignment is determined for each operation; secondly, a job shop scheduling problem is solved.

The initial assignment of the operations is given by a greedy algorithm, denoted by H. This algorithm is the same as the one described in (Dauzère-Pérès and Paulli, 1997). First, the operations are sorted in $|\mathcal{R}_{i,j}|$ non-decreasing order and by non-increasing $p_{i,j}$ order to break ties. The workload of one resource is defined by the sum of the processing times of the assigned operations. Resources are sorted in their workload non-decreasing order. For each operation $O_{i,j}$, the first resource that belong to $\mathcal{R}_{i,j}$ is assigned to the operation, the workload of this resource is updated as well as the sorting of the resources. The process iterates until all the operations are assigned to one resource.

Then, the job shop problem is solved by using the following greedy algorithm, based on a SLACK rule. We denote by $\mathcal{S}$ the set of candidate operations. At the beginning $\mathcal{S} = \{O_{1,j}, 1 \leq j \leq n\}$. The candidate operations are sorted in their release time non decreasing order (and slack non decreasing order to break ties). The release time of an operation $O_{i,j}$ is the maximum between the time the performing resource is free for processing operation $O_{i,j}$ and the completion time of operation $O_{i-1,j}$. The slack of operation $O_{i,j}$ is the difference between the due date of job J_j and the completion time of this job if the remaining operations were processed after $O_{i,j}$ without idle time. The first candidate operation is scheduled, the release dates are updated, $\mathcal{S}$ is updated and the process iterates while $\mathcal{S} \neq \emptyset$.

This initial solution tries to return a solution with a good lateness value, since due dates are considered in the SLACK rule. The reason is that the solution has to respect the bound for the lateness value, for being feasible. In the following, this constraint is the main concern.

3.2 Coding of a solution

A solution is represented by a conjunctive graph $G = (V, E)$. The set of vertices is equal to $V = \{O_{i,j}, 1 \leq j \leq n, 1 \leq i \leq n_j\} \cup \{O_{n_j+1,j}, 1 \leq j \leq n\} \cup \{s, t_L, t_C\}$ with s the source and t_L and t_C two sink nodes. There is an edge between two vertices if there is a routing constraint or a resource constraint between the corresponding operations. There is an edge between operation $O_{n_j,j}$ and operation $O_{n_j+1,j}$ of length $p_{n_j,j}$; an edge between $O_{n_j,j}$ and t_C of length $p_{n_j,j}$; and an edge between $O_{n_j+1,j}$ and t_L of length $-d_j$; an edge between s and $O_{1,j}$ of length 0. For the other edges, the length is equal to the processing time of the operation at the origin of the edge.

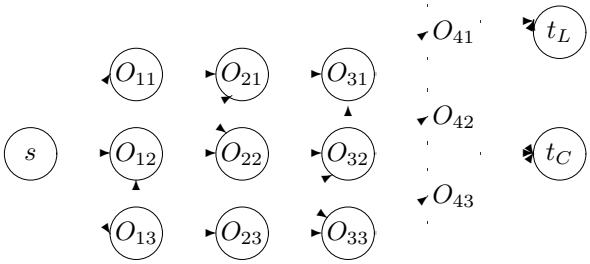

Fig. 1. Coding of a solution

The longest path between s and t_C gives the makespan of the solution and the longest path between s and t_L gives the maximum lateness.

3.3 Neighborhood

A neighbor is obtained by moving a critical operation. This move is the same as the one described in (Dauzère-Pérès and Paulli, 1997). By using this unique technique, both the modification of an assignment and the modification of a sequence can be obtained. In the conjunctive graph, moving an operation x between operations y and z means:

- deleting edge (v, x) and (x, w), where v (respectively w) is the predecessor (respectively successor) of x on the performing resource,
- adding an edge (v, w),
- deleting the edge (y, z)
- adding edges (y, x) and (x, z).

Figure 2 shows such a move in the graph.

A feasible move is a move for which operation x is assigned to a resource where it can be processed and such that no circuit is generated in the graph.

In the graph, the critical operations we consider are those for both the makespan and the maximum lateness values, that is that all the operations that belong to the longest path between s and t_C and between s and t_L.

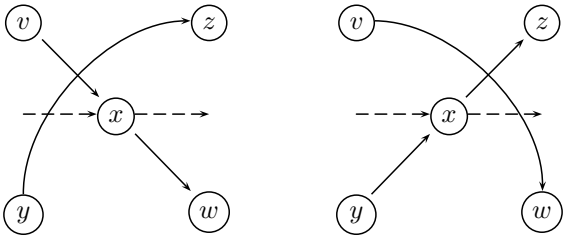

Fig. 2. Moving operation x between operations y and z

The neighborhood is defined as follows: for each critical operation, for each feasible move, a neighbor is generated.

3.4 Evaluation of a neighbor

Since we try to solve an ϵ-constraint problem, the definition of the best neighbor depends on the structure of the set of neighbors. The best neighbor of a solution is the one with the minimum C_{max} among the neighbors with $L_{max} < \epsilon$ (see case (a) in Figure 3 where the best neighbor is represented in a circle). If all the neighbors are such that $L_{max} \geq \epsilon$, then the best neighbor is the one with the minimum lateness, with a makespan value not greater than the makespan value of the current solution (in bold in Figure 3(b)) – if possible (see Figure 3(b)). In all other cases, we choose the neighbor that minimizes L_{max}.

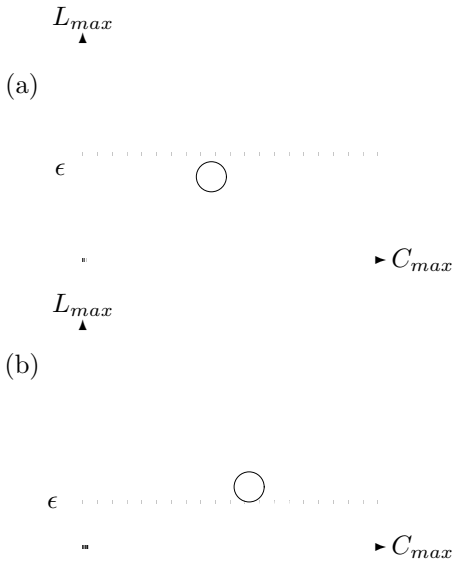

Fig. 3. Best neighbor and search direction

With this definition of the best neighbor, we first try to respect the ϵ bound and then to improve the makespan under the ϵ constraint.

3.5 The tabu list

The tabu list is used to prevent the search from cycling between solutions. Our tabu list has a fixed size, when the list is full, the oldest element is replaced by the new element.

In (Dauzère-Pérès and Paulli, 1997) the authors propose three types of tabu list. During their experiments, they found that one type of list allows to obtain the best solutions. We use this tabu list that works as follows: after moving x between y and z, (x, y) and (z, x) are added to the tabu list.

3.6 Algorithm

The Tabu search algorithm is described in Table 1. The stopping criterion is a number of iterations without improvement, fixed to $\#it_{max} = 100$ iterations. Variable *sbest* is the best solution, $s*$ is the current solution, s is a neighbor of $s*$, s' is the best neighbor of $s*$, $S'minCmax$ is the neighbor with minimum makespan among the neighbors such that $L_{max} < \epsilon$, $S'minLmaxST$ is the neighbor of $s*$ with minimum L_{max} and a makespan smaller that $C_{max}(s*)$, $S'minLmax$ is the neighbor of $s*$ with minimum L_{max} and all other neighbors have makespan greater that $C_{max}(s*)$. We denote by $\#it$ the number of iterations without improvement.

4. COMPUTATIONAL EXPERIMENTS

Four data sets have been used to evaluate the Tabu search algorithm. These data sets are based on Hurink's instances (Hurink *et al.*, 1994), derived from Lawrence's instances. In order to introduce due dates in these data sets, the method that has been used is inspired by (Demirkol *et al.*, 1998). The due date of J_j is defined by:

$$d_j \hookrightarrow \mathcal{U}\left[\mu_j \times \left(1 - \frac{R}{2}\right); \mu_j \times \left(1 + \frac{R}{2}\right)\right]$$

with

$$\mu_j = \left(1 + \frac{T \times n}{m}\right) \times \sum_{i=1}^{n_j} p_{i,j}$$

T and R are two parameters fixed to $T = 0.3$ and $R = 0.5$.

The four data sets are characterized by:

- `sdata`: instances of Lawrence (single machine per operation), no assignment problem,
- `edata`: instances with few flexibility, near to the original instances of Lawrence,
- `rdata`: instances with two possible resources per operation in average,
- `vdata`: instances with $m/2$ possible resources per operation in average.

The size of the instances la01 to la05 is ($m = 5 \times n = 10$), instances la06 to la10 (5×15), instances la11 to la15 (5×20), instances la16 to la20 (10×10), instances la21 to la25 (10×15), instances la26 to la30 (10×20), instances la31 to la35 (10×30), and instances la36 to la40 (15×15).

The bound ϵ on L_{max} is equal to $\delta L_{max}(H)$ with $\delta \in \{0.7; 0.9\}$.

Average computation times are given in Table 1. Results show that the most difficult instances are given for data sets la31 to la35 and of type `vdata`, that is when the number of possible assignments is maximum. Computation times are of the same range for $\delta = 0.7$.

The average number of iterations before respecting the bound ϵ for the first time is indicated in Table 2. Note that for some instances no solution respecting the bound ϵ has been found (see Table 5). In this case, the iterations only decrease the maximum lateness value. Results show that if the bound ϵ is small, more iterations are required, and if the flexibility increases less iterations are require. The average total number of iterations is indicated in Table 3.

Table 4 indicates the average deviation of the makespan, defined by $100 \times \frac{C_{max}(H) - C_{max}(TS)}{C_{max}(H)}$, where TS indicates the Tabou Search algorithm. In this table, we can see that the initial makespan is quite always improved, despite the fact that the solution has to respect the bound ϵ. For problem instances la10 to la15, $\delta = 0.7$ and data set `sdata`, we can see a negative average deviation. This is due to the fact that for those instances, respecting ϵ for the maximum lateness imposes decreasing the makespan value.

Table 5 indicates the number of instances for which a solution respecting the bound ϵ has been found. We can see that the more difficult instances are given by `sdata`, and `edata` if $\delta = 0.7$.

Table 6 indicates the average deviation of the makespan, defined by $100 \times \frac{C_{max}(TS') - C_{max}(DPP)}{C_{max}(TS')}$, where $C_{max}(DPP)$ indicates the makespan found in (Dauzère-Pérès and Paulli, 1997) and TS' stands for the Tabu search algorithm with $\epsilon = \infty$. In average, the deviation is never greater than 10% from the best solution, and results do not depend on the flexibility of the data set. Note that the Tabu search algorithm proposed in (Dauzère-Pérès and Paulli, 1997) cannot be used for solving the problem we consider, because of the neighborhood definition and the neighbors evaluation.

Algorithm 1 Tabu search algorithm

```
1  sbest = ∅, Cmax(sbest) = HV, Lmax(sbest) = HV
2  s*=solution returned by H
3  while #it < #itmax do
4       s' = ∅, Cmax(s') = HV, Lmax(s') = HV
5       S'minLmaxST = ∅, Cmax(S'minLmaxST) = HV, Lmax(S'minLmaxST) = HV
6       S'minLmax = ∅, Cmax(S'minLmax) = HV, Lmax(S'minLmax) = HV
7       S'minCmax = ∅, Cmax(S'minCmax) = HV, Lmax(S'minCmax) = HV
8       oneUnderϵ = false , oneBestCmax = false
9       for all non-tabu neighbor s of s* do
10           Evaluate s
11           if sbest = ∅ and Lmax(s) ≥ ϵ then
12                if Lmax(s) < Lmax(s') then s' = s end if
13           else
14                if Lmax(s) < ϵ then
15                     oneUnderϵ = true
16                     if Cmax(S'minCmax) > Cmax(s) then S'minCmax = s end if
17                else
18                     if Cmax(s) < Cmax(s*) then
19                          oneBestCmax = true
20                          if Lmax(S'minLmaxST) > Lmax(s) then S'minLmaxST = s end if
21                     else
22                          if Lmax(S'minLmax) > Lmax(s) then S'minLmax = s end if
23                     end if
24                end if
25                if oneUnderϵ == true then
26                     s' = S'minCmax
27                else
28                     if oneBestCmax == true then s' = S'minLmaxST
29                     else s' = S'minLmax end if
30                end if
31           end if
32      end for
33      s* = s'
34      if oneUnderϵ == true and Cmax(sbest) > Cmax(s') then sbest = s', #it = 0 else #it = #it + 1 endif
35      Update tabu list
36 end while
37 return sbest
```

5. CONCLUSION AND FURTHER DIRECTIONS

In this paper, we have investigated a flexible job shop scheduling problem with two objective measures: the makespan and the maximum lateness. We have proposed a Tabu search algorithm that solves the ϵ-constraint formulation of this bicriteria problem. Instances adapted from the literature have been tested and results show that the method solves the problems in a reasonable computation time and gives solutions of relatively good quality.

One interesting future research direction consists in improving the initial solution, so as to study its impact on the final solution. Then, this method will be used for obtaining a set of solutions with non-dominated criteria vectors (trade-off curve approximation).

REFERENCES

Alvarez-Valdes, R., A. Fuertes, J. M. Tamarit, G. Giménez and R. Ramos (2005). A heuristic to schedule flexible job-shop in glass factory. *European journal of operational research* **165**, 525–534.

Dauzère-Pérès, S. and J. Paulli (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Resarch* (70), 281–306.

Demirkol, E., S. Mehta and R. Uzsoy (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research* **109**, 137–141.

Glover, F. (1989). Tabu search - part i. *ORSA Journal on Computing* **1**, 190–206.

Hurink, J., B. Jurisch and M. Thole (1994). Tabu search for the job-shop scheduling problem with multipurpose machines. *OR Spektrum* **15**, 205–215.

Jurisch, B. (1992). Scheduling Jobs in Shops with Multi-Purpose Machines. PhD thesis.

Mastrolilli, M. and L-M. Gambardella (2000). Effective neighborhood function for the flexible job shop problem. *Journal of Scheduling* **3**(1), 3–20.

Paulli, J. (1995). A hierarchical approach for the fms scheduling problem. *European Journal of Operational Research* **86**, 32–42.

T'kindt, V. and J-C. Billaut (2002). *Multicriteria scheduling: theory, models, algorithms.* Springer. Berlin.

Table 1. Average computation times of the Tabu search algorithm (in sec.) for $\delta = 0.9$.

data	sdata	edata	rdata	vdata
la01 to la05	0.87	0.77	1.58	2.75
la06 to la10	1.04	2.97	4.03	6.97
la10 to la15	2.49	4.56	11.94	12.60
la16 to la20	1.15	1.14	3.07	5.84
la21 to la25	4.54	5.97	17.28	28.61
la26 to la30	9.69	12.00	39.97	106.21
la31 to la36	2.79	38.50	131.82	501.39
la36 to la40	4.91	10.60	21.44	109.05

Table 2. Average number of iterations before respecting ϵ.

		$\delta = 0.9$		
data	sdata	edata	rdata	vdata
la01 to la05	49.6	2.6	1.4	1.4
la06 to la10	69.8	22.0	15.0	1.2
la10 to la15	104.2	45.4	4.8	6.8
la16 to la20	36.8	3.0	13.0	2.0
la21 to la25	17.2	27.6	2.8	1.8
la26 to la30	20.0	17.6	6.4	3.2
la31 to la36	54.4	11.2	6.8	10.2
la36 to la40	63.6	4.6	2.6	5.4
		$\delta = 0.7$		
data	sdata	edata	rdata	vdata
la01 to la05	77.0	61.2	6.8	4.8
la06 to la10	131.0	88.2	45.0	12.8
la10 to la15	127.2	117.6	138.6	78.4
la16 to la20	97.6	50.6	6.8	6.4
la21 to la25	160.6	139.4	11.8	13.2
la26 to la30	236.4	144.0	33.4	15.8
la31 to la36	188.6	279.4	96.2	69.4
la36 to la40	183.6	78.8	18.6	19.2

Table 3. Average total number of iterations.

		$\delta = 0.9$		
data	sdata	edata	rdata	vdata
la01 to la05	149.4	122.6	190.2	287.8
la06 to la10	110.2	244.6	196.6	271.2
la10 to la15	127.2	206.6	252.4	218.2
la16 to la20	130.6	118.0	201.0	181.4
la21 to la25	223.2	264.2	471.0	310.0
la26 to la30	240.6	273.0	492.2	538.2
la31 to la36	242.8	322.0	490.0	624.2
la36 to la40	156.8	294.4	354.2	451.0
		$\delta = 0.7$		
data	sdata	edata	rdata	vdata
la01 to la05	116.8	130.8	144.0	224.2
la06 to la10	130.0	132.0	149.8	161.0
la10 to la15	126.2	181.2	190.2	214.0
la16 to la20	139.0	139.0	202.6	255.8
la21 to la25	180.2	163.0	297.2	397.4
la26 to la30	235.4	231.0	611.6	555.8
la31 to la36	187.6	278.4	521.4	471.4
la36 to la40	182.6	164.2	282.8	406.6

Table 4. Average makespan deviation.

		$\delta = 0.9$		
data	sdata	edata	rdata	vdata
la01 to la05	11.9%	10.2%	24.6%	21.7%
la06 to la10	1.6%	13.2%	18.4%	18.0%
la10 to la15	1.1%	6.9%	12.7%	14.7%
la16 to la20	3.4%	9.7%	20.5%	26.7%
la21 to la25	10.9%	12.9%	26.6%	26.8%
la26 to la30	6.6%	13.7%	31.3%	29.6%
la31 to la36	6.7%	12.4%	22.1%	25.2%
la36 to la40	4.5%	14.1%	24.4%	28.1%
		$\delta = 0.7$		
data	sdata	edata	rdata	vdata
la01 to la05	9.0%	8.1%	19.1%	19.0%
la06 to la10	2.5%	6.1%	14.5%	13.0%
la10 to la15	−0.4%	3.5%	6.4%	10.1%
la16 to la20	2.5%	10.6%	19.6%	28.2%
la21 to la25	6.9%	4.6%	25.8%	27.6%
la26 to la30	3.8%	9.4%	29.5%	29.4%
la31 to la36	4.9%	7.2%	19.2%	22.4%
la36 to la40	0.2%	6.1%	20.4%	26.0%

Table 5. Number of instances for which the bound is respected.

		$\delta = 0.9$		
data	sdata	edata	rdata	vdata
la01 to la05	**3**	5	5	5
la06 to la10	**2**	**4**	5	5
la10 to la15	**1**	**4**	5	5
la16 to la20	**4**	5	5	5
la21 to la25	5	5	5	5
la26 to la30	5	5	5	5
la31 to la36	**4**	5	5	5
la36 to la40	**3**	5	5	5
		$\delta = 0.7$		
data	sdata	edata	rdata	vdata
la01 to la05	**2**	**3**	5	5
la06 to la10	**0**	**2**	**4**	5
la10 to la15	**0**	**2**	**2**	**3**
la16 to la20	**2**	**4**	5	5
la21 to la25	**1**	**1**	5	5
la26 to la30	**0**	**3**	5	5
la31 to la36	**0**	**0**	5	5
la36 to la40	**0**	**3**	5	5

Table 6. Average makespan deviation with DPP

data	sdata	edata	rdata	vdata
la01 to la05	4.8%	8.9%	3.2%	2.6%
la06 to la10	2.5%	3.5%	1.7%	2.8%
la10 to la15	3.7%	7.4%	2.7%	1.9%
la16 to la20	7.7%	6.9%	10.9%	5.2%
la21 to la25	10.9%	9.4%	10.8%	10.8%
la26 to la30	13.9%	11.7%	11.1%	6.8%
la31 to la36	10.3%	10.9%	7.0%	4.5%
la36 to la40	11.3%	14.8%	16.4%	11.2%