

# Order Acceptance and Scheduling Decisions in Make-to-Order Systems

Zehra Bilgintürk, Ceyda Oğuz, Sibel Salman

Koç University, Rumelifeneri Yolu, 34450, Sarıyer, Istanbul, Turkey, {zbilginturk, coguz, ssalman}@ku.edu.tr

---

In this paper, we examine simultaneous order acceptance and scheduling decisions in a make-to-order system. We model the manufacturing environment as a single machine environment, with a set of orders coming from customers. In this pool of orders, of which we know the release dates, due dates, processing times, deadlines, sequence dependent setup times and revenues, manufacturer has to decide on the subset of orders to select, considering the limited production capacity in order to maximize the profit. The tardiness of orders is penalized and revenue gained from an order decreases with tardiness, until the deadline, after which no revenue can be obtained. The problem generalizes some well-known scheduling problems with the objective of minimizing total tardiness and has the sequence dependent setup times as a further complicating property, as well as the order acceptance decisions. As a result, the problem is NP-hard. In this study, first, we give an MILP model for the problem, and solve it with a commercial solver. Next, we propose a Simulated Annealing algorithm to solve the large-size problems. We compare the performance of the algorithm with respect to the optimal solution, when the problem can be solved optimally. We use upper bounds generated by LP relaxation for comparison when optimal objective values are not available.

*Keywords:* Production Scheduling, Meta-heuristic Search.

---

## 1 Introduction

Make-to-order firms, such as a custom-made furniture shop or a boat builder, manufacture customized products, in which the production is initiated by a customer order and, typically, no work-in-process inventory is carried for this kind of products. The manufacturer will gain a revenue if a particular order is accepted and manufactured by using the production resources. Since the main production resource, i.e., the capacity is limited, manufacturing one order may delay another one. If there are no deadlines, then the manufacturer will just incur some penalty for the orders which are delayed. If there are deadlines for the orders, such delays may lead to rejecting customer orders. In a competitive make-to-order environment, a manufacturer should use the capacity efficiently, satisfy the customers' expectations at a high level and gain the maximum revenue from the incoming orders. Therefore, the manufacturer should find a balance between the revenue that can be gained from the accepted orders and the cost incurred by manufacturing these orders. So the question is which orders to accept to maximize the profit considering the limited production capacity of the firm. Accepting an order whose deadline cannot be met causes a loss in the reputation of the firm as well as in the revenue. To avoid such circumstances, the order acceptance decisions should be made very carefully. When high utilization levels are present, firms accept the orders that will bring high revenues and have comparably less production capacity requirements only. If an order will be tardy, that is, if its completion time will exceed its due date, the manufacturer has to sacrifice from some of its revenue, if this order is accepted.

In this study we consider the problem from the manufacturer's point of view and try to maximize the profit gained from the accepted orders. The profit is defined as the total revenues minus the total weighted tardiness. The manufacturing environment consists of a single machine with a

limited capacity and a set of independent orders  $O$  ( $|O| = n$ ) at the beginning of the planning period. For each order  $i \in O$ , we know the data including the release date  $r_i$  after which the order is available, the processing time  $w_i$ , the due date  $d_i$  up to which order  $i$  can be produced without incurring a penalty, the deadline  $\bar{d}_i$  after which order  $i$  cannot be produced ( $\bar{d}_i \geq d_i$ ), the sequence dependent setup time  $s_{ij}$  which is required when order  $i$  is scheduled before order  $j$ , the revenue  $e_i$  to be gained if the order is accepted. The revenue  $e_i$  may also reflect the level of priority or the importance of order  $i$ . The decision to be made in this environment includes determining the set of accepted orders and the corresponding optimal schedule for this set of orders to minimize the weighted tardiness. The tardiness of order  $i$  is defined as  $T_i = \max\{0, C_i - d_i\}$  where  $C_i$  is the completion time of order  $i$ .

As our aim is to maximize the profit gained from the accepted orders, which is defined as the total revenues minus the total weighted tardiness, our objective function is then defined by  $\max \sum_{i \in A} P_i = \sum_{i \in A} (e_i - T_i \times \alpha_i)$  where  $A$  is the set of accepted orders,  $\alpha_i$  is the weight for the tardiness of order  $i$  and  $P_i$  is the profit gained from the accepted order  $i$ . The profit  $P_i$  reflects the fact that if an order  $i$  is tardy, the customer will receive a discount. However we note that the early delivery is not penalized as the manufacturer is assumed to have unlimited capacity for finished product storage. In this setting, the manufacturer has the right to reject an order that will not be profitable. We will denote this problem as OASP (Order Acceptance and Scheduling Problem) throughout the paper.

OASP is a common real-life problem faced at the production companies as well as the service companies. One such example to manufacturing companies facing this problem is a packaging firm that has varying sequence dependent setup times for each order. Consulting firms is an example of service companies working on a make-to-order basis.

The relevant research to OASP can be categorized under two subtitles: 1) studies concerning the total tardiness problem, since OASP is a generalization of the single machine total tardiness problem, and 2) studies concerning the simultaneous order acceptance and scheduling problem.

The total tardiness problems have been studied since 1960s. Emmons[1] is one of the first researchers working on the total tardiness problems. Du and Leung [2] proved that minimizing total tardiness in one machine is NP-hard in the ordinary sense. The addition of sequence dependent setup times increases the veracity of the problem, but it increases the computational complexity as well. Hence, the total tardiness problem with sequence dependent setup times (TTSDS) is also NP-hard. Comprehensive literature reviews on setup considerations in scheduling problems are presented by Allahverdi et al. [3] and Yang et al. [4].

Application of metaheuristics as well as branch-and-bound algorithm to the TTSDS is common. Rubin and Ragatz [5] applied a genetic search to the problem. The success of simulated annealing technique in combinatorial optimization problems motivated researchers to apply this method to TTSDS problem. Tan and Narasimhan [6] used a Simulated Annealing approach to TTSDS problem. The algorithm proposed by the authors is not suitable for OASP since OASP contains additional complicating issues such as the order acceptance, the release dates, and the deadlines.

A recent study on order acceptance and scheduling problem is presented by Slotnick and Morton [7]. OASP is a generalization of this problem as it takes the sequence dependent setup times into account. Also OASP problem fills a gap in the scheduling literature since make-to-order environments with the sequence dependent setup times and strict deadlines are not covered.

In the remaining parts of the paper, we will first present an MILP model for OASP. As our computational experiments showed that it is not possible to solve this model with a branch-and-bound algorithm if  $n > 10$ , we then propose a Simulated Annealing algorithm to solve the problem with large-size instances. We conclude the paper with the results of our computational experiments

in which the performance of the Simulated Annealing algorithm is analyzed.

## 2 Mathematical Model

We formulate the order acceptance and sequencing problem as a mixed integer problem below. In this model, we define dummy orders, order 0 and order  $n + 1$ , where definitely, order 0 is assigned to the first position, and order  $n + 1$  is assigned to the last position, whatever the sequence of orders in between is. There are two sets of binary variables in the model. The first one,  $y_{ij}$ , determines the sequence of orders, and the second one,  $I_i$ , determines which orders are accepted. Hence,  $y_{ij} = 1$  if order  $i$  precedes order  $j$ , and  $y_{ij} = 0$  otherwise; and  $I_i = 1$  if order  $i$  is selected, and  $I_i = 0$  otherwise. The objective function of the model is to maximize the total profit gained from accepted orders.

Constraint sets (1) and (2) states that, if an order is accepted, this order precedes only one job and, is succeeded by only one job. If it is not accepted, it does not take place in the sequence. These constraints also prohibit the preemption of the jobs. Constraint set (3) implies that, if order  $j$  is preceded by order  $i$ , then the completion time of order  $j$  should be larger than the completion time of order  $i$  plus the sequence dependent setup time between orders  $i$  and  $j$ , plus the processing time of order  $j$ . If order  $i$  does not precede order  $j$  in the sequence,  $C_j \geq 0$  should be the only limit, and this constraint ensures this outcome. Constraint set (4) states that, if order  $j$  is accepted, and is preceded by order  $i$  in the schedule, then the completion time of order  $j$  should be greater than the release date of that order plus the sequence dependent setup time between orders  $i$  and  $j$ , plus the processing time of order  $j$ . In case where order  $i$  does not precede order  $j$ , the completion time of order  $j$  has looser bounds which are given by the release time of order  $j$  and its deadline. In case where order  $j$  is not accepted, it will not be processed so,  $C_j = 0$ . These constraints enable us to calculate the correct completion times of the orders. Constraint sets (5) and (6) are very general constraints and they put bounds on  $C_i$  and  $T_i$ . Set (7) calculates the maximum tardiness for an accepted order. Sets (8) and (10) ensure nonnegativity of  $P_i$  and  $T_i$ , implied by the definitions of these decision variables. Constraint set (9) calculates the profit manufacturer can gain, when order  $i$  is accepted and incurs tardiness of  $T_i$ . Here, the profit from an accepted order decreases as this order becomes tardy and becomes 0 at its deadline.  $\alpha_i$  coefficient is chosen accordingly to satisfy this assumption. For one unit time of tardiness, revenue decreases by  $\alpha_i$  units.  $\alpha_i$  coefficient is calculated by the formula  $e_i/(\bar{d}_i - d_i)$  for each order  $i$ . Sets (11) and (12) include some necessary initializations for dummy nodes 0 and  $n + 1$ . Constraint set (13) defines the decision variables of the model as binary variables.

When we solve this MILP by CPLEX solver, it is observed that the optimal solution can be obtained only up to  $n = 10$  orders, which is not surprising. Hence we resort to metaheuristic algorithms as they were successfully applied to TTSDS problem. In this study we chose Simulated Annealing and analyzed its performance.

**MILP:**

$$\max \sum_{i=1}^n P_i$$

$$s.t \quad \sum_{j=1, j \neq i}^{n+1} y_{ij} = I_i, \quad \forall i = 0, \dots, n \quad (1)$$

$$\sum_{j=0, j \neq i}^n y_{ji} = I_i, \quad \forall i = 1, \dots, n+1 \quad (2)$$

$$C_i + (s_{ij} + w_j) \times y_{ij} + \bar{d}_i \times (y_{ij} - 1) \leq C_j, \quad \forall i = 0, \dots, n, \forall j = 1, \dots, n+1, i \neq j \quad (3)$$

$$r_j \times I_j + (s_{ij} + w_j) \times y_{ij} \leq C_j, \quad \forall i = 0, \dots, n, \forall j = 1, \dots, n+1, i \neq j \quad (4)$$

$$T_i \geq C_i - d_i, \quad \forall i = 0, \dots, n+1 \quad (5)$$

$$C_i \leq \bar{d}_i \times I_i, \quad \forall i = 0, \dots, n+1 \quad (6)$$

$$T_i \leq (\bar{d}_i - d_i) \times I_i, \quad \forall i = 0, \dots, n+1 \quad (7)$$

$$T_i \geq 0, \quad \forall i = 0, \dots, n+1 \quad (8)$$

$$P_i \leq e_i \times I_i - T_i \times \alpha_i, \quad \forall i = 1, \dots, n \quad (9)$$

$$P_i \geq 0, \quad \forall i = 1, \dots, n \quad (10)$$

$$C_0 = 0, C_{n+1} = \max_{i=1, \dots, n} \{\bar{d}_i\}, \quad \forall i = 1 \dots n \quad (11)$$

$$I_0 = 1, I_{n+1} = 1 \quad (12)$$

$$I_i \in \{0, 1\}, y_{ij} \in \{0, 1\}, \quad \forall i = 1, \dots, n \quad (13)$$

### 3 Simulated Annealing Algorithm

In this section we describe the Simulated Annealing algorithm for solving the OASP. Simulated Annealing is a heuristic method that uses thermodynamic concepts. Simulated Annealing has a very important aspect in its nature: it is good at diversifying and less likely to get stuck in the local optimum than other metaheuristics. Simulated Annealing accepts a new solution even it is worse than before, with a probability given by  $\exp(-\delta f/T)$ , where  $\delta f$  is the increase in the objective function value and  $T$  is the current temperature. Simulated Annealing technique was successfully applied to combinatorial optimization problems in the literature as in [6]. Our algorithm is described below.

1. Read the input data.
2. Set the control parameters:
  - (a) Initial temperature ( $T_{max}$ )
  - (b) Temperature decay rate,  $\theta$
  - (c) Set  $T_{current} = T_{max}$
3. Construct the initial solution (not necessarily feasible).
  - (a) Sequence the jobs in descending order of  $slacktime_j = \bar{d}_j - r_j - (minsetup_j + w_j)$ ;
  - (b) Calculate the completion time  $C_j$ , the tardiness  $T_j$  and the revenue gained,  $P_j$ .

- (c) Count the number of orders violating their deadlines  $d_j$ .
4. While number of violating orders  $> 0$  do the following:
- (a) Perform the following loop ITER times:
    - i. Generate two different random integers between 1 and  $n$ . ( $n$  is the number of available orders)
    - ii. Exchange the orders having these indices.
    - iii. Calculate  $C_j$ 's,  $T_j$ 's and  $P_j$ 's and the number of violating orders in this new sequence.
      - A. if (Total revenue  $>$  best revenue), accept the new sequence.
      - B. if (Total revenue  $\leq$  best revenue),
        - Calculate the probability of accepting,  $p = \exp(-(-\text{Total revenue} + \text{best revenue})/T_{\text{current}})$ .
        - Select uniformly distributed random number  $m$ , from the interval (0,1).
          - if  $m < p$ , accept the sequence.
          - if  $m \geq p$  reject the new sequence, and continue with the former best sequence.
      - C. Return to Step (i).
  - (b) Set  $T_{\text{current}} = \theta \times T_{\text{current}}$
  - (c) Calculate the ratio  $ratio_j = e_j / (w_j + \text{minsetup}_j)$  for violating orders.
  - (d) Reject the order having the smallest ratio.
  - (e) Return to Step (4) with new sequence.
5. If number of violating orders = 0, i.e., a feasible solution is obtained, perform the following for ITER1 times:
- (a) Generate two different random integers between 1 and  $n$ . ( $n$  is the number of available orders)
  - (b) Exchange the orders having these indices.
  - (c) Calculate  $C_j$ 's,  $T_j$ 's and  $P_j$ 's and the number of violating orders in this new sequence.
    - Calculate the probability of accepting,  $p = \exp(-(-\text{Total revenue} + \text{best revenue})/T_{\text{current}})$ .
    - Select uniformly distributed random number  $m$ , from the interval (0,1).
      - if  $m < p$ , accept the sequence.
      - if  $m \geq p$  reject the new sequence, and continue with the former best sequence.

## 4 Computational Results

In order to analyze the performance of the Simulated Annealing algorithm (SA), we performed a computational experiment. In this section we first describe the data generation together with the experiments and then present the results.

In this study, release dates  $r_i$ , processing times  $w_i$ , sequence dependent setup times  $s_{ij}$ , and revenues  $e_i$  are generated from a uniform distribution from the interval of  $[0, w_T \times \tau]$ ,  $[1, 20]$ ,  $[1, 10]$ , and  $[1, 20]$ , respectively. The due dates are generated so that for each order  $i$ , due date  $d_i$  is in the interval of  $(w_T + \max_j s_{ij} + r_i) \times [1 - \tau - R/2, 1 - \tau + R/2]$  and integer, where  $\tau$ ,  $R$  and  $w_T$  represent the tardiness factor, the due date range and the total processing time of all orders respectively. The deadlines are generated from the formula  $\bar{d}_i = d_i + R \times w_i$ . The weight  $\alpha_i$  is calculated by the

formula  $\alpha_i = e_i / (\bar{d}_i - d_i)$ . In this setting, revenue gained from an order  $i$  becomes 0 at its deadline  $\bar{d}_i$ .

The first part of the computational study compares the performance of SA with respect to the optimal objective values. The MILP model was coded in ILOG and solved by CPLEX solver. Time limit was 5000 seconds for the CPLEX runs. The SA was coded in JAVA. All of the runs were taken on a computer with a Pentium 4 processor, 2.4 GHz speed, and 2 GB of RAM. We present the maximum, minimum, and average % deviation of the SA objective function value from the optimal objective function value and the corresponding CPU times in Table 1. For each parameter combination, we solved 5 instances of OASP, generated randomly as explained above, and reported the results. From these results, it is seen that the hardest problem instance for CPLEX is generated when  $\tau = 0.1$  and  $R = 0.1, 0.3$ . For the problem sizes above 10, it is impossible to find the optimal solution within the time limit of 5000 seconds in this setting. The results of the experiments confirm that, as  $\tau$  increases, the problem gets less time consuming for the optimal method, but more time consuming for the SA. In addition, when we keep  $\tau$  constant, and increase  $R$ , we see that the problem becomes easier for the optimal method. We found that the SA objective function values are competitive with the optimal objective function values for  $n = 10$  which can be observed from the average percentage deviations displayed in Table 1. SA algorithm is clearly a fast and a good quality metaheuristic for  $n = 10$ .

Table 1: SA and optimal solution comparison for the OASP with problem sizes of 10.

			% Deviation			CPU Time (SA)			CPU Time (CPLEX)		
n	$\tau$	R	Max	Min	Avg	Max	Min	Average	Max	Min	Average
10	0.1	0.1	6%	0%	2%	35.62	30.98	32.36	> 5000.00	> 5000.00	> 5000.00
		0.3	10%	2%	6%	42.06	32.41	34.46	> 5000.00	> 5000.00	> 5000.00
		0.5	0%	0%	0%	51.08	22.94	29.85	> 5000.00	190.43	2199.06
		0.7	0%	0%	0%	23.84	22.98	23.45	2101.82	7.82	426.38
		0.9	0%	0%	0%	39.39	23.16	26.78	3545.00	2.00	1314.93
	0.3	0.1	11%	0%	4%	136.67	47.98	104.94	> 5000.00	1430.73	3569.27
		0.3	12%	3%	6%	199.60	73.57	118.18	> 5000.00	2390.90	3546.89
		0.5	4%	0%	2%	130.41	77.13	110.41	3111.22	878.61	2027.17
		0.7	8%	0%	2%	83.93	23.35	45.15	> 5000.00	58.29	2067.75
		0.9	1%	0%	0%	38.89	23.17	29.71	2419.26	35.67	932.12

As it is not possible to obtain the optimal solution for OASP when  $n > 10$ , we compared the performance of the SA with respect to an upper bound in the second part of the computational study. An easy to compute upper bound is found by the LP relaxation of MILP. For each parameter combination of the problem, we solved 10 instances generated randomly as described above and reported the results. We present the maximum, minimum, and average % deviation of the SA objective function value from the objective function value of the LP relaxation and the corresponding CPU times in Table 2. We carried out the experiments for problem sizes of 10, 15, 20, 25, and 50. LP relaxation is observed to be relatively tight when tardiness factor  $\tau$  is equal to 0.1 and 0.3 for  $n=10$  (average SA-LP gaps are around 1% and 9%, respectively). LP relaxation solution times are not included in the table since the problem is solved within maximum of 3–4 seconds when integrality restrictions are discarded.

We can make the following observations from Table 2:

- The CPU times of SA are reasonable even for large-size problem instances.
- The average % deviation is below 25% in all cases.

- The observation we made for small-size problem instances regarding the difficult problem cases is not clear as in Table 1 because when we analyze the CPU times of SA, we cannot see much difference depending on different cases. However, we observe that the average % deviation of the difficult cases for small-size problems is large in Table 1 as well. When we examine the results in Table 2, we see that the differences among the average % deviations for different cases are not so large. Hence, we may conclude that for large-size problems, the difficulty of the problem remains the same for different  $R$  values.
- We observe that as  $\tau$  increases, the problem gets more difficult for SA.
- The large % deviations can be attributed to the difficulty of the problem, which results in loose upper bounds from LP relaxations.

Overall we can say that SA is an acceptable solution method for OASP even for large-size problems as its average % deviation from the LP relaxation is below 25%.

## 5 Summary and Conclusions

In this study, we have studied a problem of simultaneous order acceptance and scheduling decisions in a make-to-order system and have provided preliminary results obtained from the computational experiments. First, we developed an MILP model for the problem. We obtained the optimal solutions for small-size problems (i.e,  $n$  is up to 10) and observed that when the problem size is greater than 10, the problem becomes intractable in a reasonable time limit. For this reason, we then proposed a Simulated Annealing algorithm for the problem. The results of the computational experiments showed that the Simulated Annealing algorithm finds good solutions compared to upper bounds which were found by LP relaxation in reasonable time limits for large-size problems.

As the next step of our study, we will continue the computational analysis and investigate the difficulty of the problem for other values of  $\tau$ .

## References

- [1] H. Emmons (1969), One machine sequencing to minimize the certain functions of job tardiness, *Operations Research* **17**, 701 – 715.
- [2] J. Du and J. Leung (1990), Minimizing total tardiness on one machine is NP-Hard, *Mathematics of Operations Research* **15**, 483 – 495.
- [3] A. Allahverdi, J.N.D. Gupta and T. Aldowaisan (1999), A review of scheduling research involving setup considerations, *Omega* **27**, 219 – 239.
- [4] W. Yang and C. Liao (1999), Survey of scheduling involving setup times, *International Journal of Systems Science* **30**, 143 – 155.
- [5] P.A. Rubin , G.L. Ragatz (1995), Scheduling in a sequence dependent setup environment with genetic search, *Computers and Operations Research* **22**, 85 – 99.
- [6] K.C. Tan and R. Narasimhan (1997), Minimizing tardiness on a single processor with sequence dependent setup times: A simulated annealing approach, *Omega* **25**, 619 – 634.
- [7] S.A. Slotnick and T.E. Morton (2007), Order acceptance with weighted tardiness, *Computers and Operations Research* **forthcoming**.

Table 2: SA and LP relaxation comparison for the problem with different sizes.

n	$\tau$	R	% Deviation			CPU Time (SA)		
			Max	Min	Avg	Max	Min	Average
10	0.1	0.1	7%	1%	3%	79.84	30.98	43.77
		0.3	10%	0%	3%	42.86	32.41	42.11
		0.5	3%	0%	0%	51.08	22.94	35.77
		0.7	0%	0%	0%	41.78	22.98	33.66
		0.9	5%	0%	1%	41.69	23.16	34.20
	0.3	0.1	25%	2%	15%	136.67	47.98	97.44
		0.3	24%	8%	14%	199.60	63.69	98.81
		0.5	17%	1%	7%	130.41	60.91	92.65
		0.7	13%	0%	6%	83.93	23.35	50.33
		0.9	16%	0%	3%	61.61	23.17	37.95
15	0.1	0.1	15%	3%	8%	109.93	83.62	90.56
		0.3	16%	1%	5%	87.48	63.53	75.61
		0.5	6%	0%	3%	82.59	45.71	63.00
		0.7	3%	0%	1%	87.06	45.48	55.81
		0.9	10%	0%	2%	106.48	45.78	60.96
	0.3	0.1	21%	10%	16%	154.93	117.27	140.36
		0.3	23%	4%	14%	151.39	105.89	126.02
		0.5	17%	6%	11%	126.48	90.42	108.72
		0.7	19%	3%	10%	127.95	85.46	109.74
		0.9	16%	2%	9%	145.36	66.11	106.40
20	0.1	0.1	11%	3%	7%	140.92	114.92	126.51
		0.3	7%	1%	4%	119.35	93.17	103.63
		0.5	4%	0%	2%	118.21	56.85	85.63
		0.7	5%	0%	2%	113.89	51.82	76.85
		0.9	10%	0%	3%	116.39	49.00	73.43
	0.3	0.1	26%	10%	18%	209.56	149.82	190.16
		0.3	26%	11%	16%	204.43	154.21	181.14
		0.5	19%	6%	14%	230.00	144.76	167.63
		0.7	18%	6%	14%	203.40	139.40	160.72
		0.9	29%	6%	13%	182.89	136.95	154.27
25	0.1	0.1	15%	5%	8%	197.95	143.79	179.01
		0.3	9%	3%	6%	172.50	124.53	146.98
		0.5	9%	2%	5%	106.87	139.86	122.92
		0.7	7%	0%	4%	54.77	125.03	105.93
		0.9	8%	0%	3%	133.92	76.72	99.20
	0.3	0.1	23%	11%	19%	288.22	243.44	260.58
		0.3	21%	13%	17%	288.42	211.38	246.88
		0.5	23%	8%	15%	265.49	197.74	231.86
		0.7	31%	9%	16%	263.27	196.89	229.27
		0.9	28%	7%	14%	289.56	199.17	228.99
50	0.1	0.1	12%	7%	10%	555.13	413.41	499.36
		0.3	10%	5%	8%	527.08	383.69	455.88
		0.5	13%	2%	9%	492.39	337.47	433.54
		0.7	14%	6%	8%	516.08	345.20	430.44
		0.9	12%	4%	10%	591.00	326.92	434.34
	0.3	0.1	28%	19%	25%	854.69	669.08	771.91
		0.3	28%	16%	21%	907.72	683.11	747.74
		0.5	25%	17%	22%	794.33	690.00	757.54
		0.7	29%	19%	23%	847.30	709.92	785.50
		0.9	33%	17%	25%	824.16	642.45	759.00