Modeling and Solving the Open-End Bin Packing Problem

Maiza Mohamed and Tebbal Mohamed and Rabia Billal
Laboratoire de Mathématiques Appliquées
Ecole Militaire Polytechnique
B.P. 17 Bordj-El-Bahri
Alger, 16111 Algérie

***Abstract*—In the Open-End Bin Packing Problem a set of items with varying weights must be packed into bins of uniform weight limit such that the capacity of the bin can be exceeded only by the last packed item, known as the *overflow item*. The objective is to minimize the number of used bins. In this paper, we present our Integer Linear Program model based on a modification of Cesili and Righini model [1]. Also, we propose two greedy heuristics to solve a problem. The first one is an adaptation of the *Minimum Bin Slack* heuristic where we have reduced to one unit capacity, the weight of the largest item in the current bin. While, the second heuristic is based on the well-known *First Fit Decreasing* heuristic. Computational results based on benchmark instances taken from the literature as well as generated instances show the effectiveness of the proposed heuristics in both solution quality and time requirement.**

***Keywords*—Open-End Bin-packing; heuristics; discrete optimization; combinatorial problem**

I. Introduction

The one dimensional open-end bin packing problem (OEBPP) is a variant of the classical bin packing problem. In the OEBPP, items with varying weights are packed into identical bins such that in each bin, the total items weight content before packing the last item is strictly less than the bin capacity. The aim of the OEBPP is to minimize the number of bins used to pack all items.

This problem was introduced by Leung et al. [2], where the authors proved that the OEBPP is NP-hard. Various application of this problem can be found in a wide variety of industries such as manufacturing, transportation, affectation tasks, etc. For example, in the fare payment system in the Hong Kong and Taipei subway stations, the passengers can buy a ticket of a fixed value. A machine at the entrance gateway records the ticket's value, while another machine deducts the fare from the ticket's value at the exit gateway. The machine at the exit gateway returns the ticket if the remaining balances of the ticket is positive; otherwise, it keeps the ticket. The objective from a passenger's point of view is to minimize the number of tickets they need to purchase as to minimize the number of bins generated by an algorithm. Another application can be found in job scheduling in the manufacturing environment that operates one shift per day. A worker loads jobs to a machine that can operate automatically. The objective is to schedule jobs to minimize the total time (days) to complete all the jobs. Intuitively, the job with the longest processing time is scheduled to the last to fully utilize the automated machine when the worker leaves from work, and the worker can unload the finished job next day.

Whilst, a few work carried out on this problem, we present a modification of the 0-1 linear programming formulation proposed by Ceselli and Righini [1] in which the authors studied a variant of the open-end bin packing called Ordered Open-End Bin Packing Problem (OOEBBP). Then we present our contributions based on the proposition of two greedy heuristics for solving the problem in an offline mode while ensuring high quality solutions in very short computational time compared to solving the problem in optimal way.

The remainder of this paper is organized as follows. In the next Section, we present a few existing lower and upper bounds on which we based to develop our propositions. Suitable way to formulate the problem with an Integer Linear Program formulation without considering the order between items is presented in Section III. In Section IV, we describe and develop our proposed heuristics which are tested and compared by means of computational tests on benchmark instances in Section V. The last Section provides some final conclusions and directions for future work.

In the next, we assume the following notations:

C:	Bin capacity;
I:	Set of item;
i;j:	Index of items and bins respectively;
w_i:	Weight or size of item i;
r_j:	Residual capacity of bin j;
S:	Set of candidate items;
O:	Set of overflow items;
R:	Overall residual capacity of used bins;

II. Literature Review

While there exists abundant research for the classical bin packing problem (interested reader can reffered to the chapter of Coffman et al. [3] for an excellent survey of this topic), there is much less research on the OEBPP. Initially, Leung et al. [2] proposed this variant of bin packing, in which the authors modelled the ticketing system at the subway station in Hong Kong. The authors was showed that any online algorithm for the disscused problem have an asymptotic worst-case ratio of at least 2. Thereafter, Yang and Leng [4] studied the Ordered Open Bin Packing Problem which extends the requirement corresponding to the order of passenger's itinerary in subway station problem. The same problem, which is the Ordered Open-End Bin Packing Problem, have been also studied by Cesili and Righini [1] where authors present branch and price algorithm for its exact optimization.

A. Lower bounds

In this section we briefly describe two existing lower bounds from the OEBPP literature.

Ongkunaruk's lower bound: Ongkunaruk [5] presented a simple idea used to compute a lower bound for the OEBPP, it consists in determining the number of needed overflow items firstly in order to apply the continuous lower bound for the rest of items. The overflow items are those selected to be packed as the last item which exceed the capacity of bin. Generally, overflow items have a large weight.

The main target in the OEBPP is to maximize the overflow items in order to minimize the capacity of items inside bins, so minimize the number of bins used.

For do, the author order items in the non-increasing size firstly, then determine the smallest integer K which indicate the number of overflow item, using the following formula:

$$\sum_{j=K+1}^{n} w_j \leq K\dot{C} \tag{1}$$

So the lower bound of Ongkunaruk L^0_{OBP} can be expressed by:

$$L^0_{OBP} = K \tag{2}$$

Ceselli's lower bound: Ceselli and Righini [1] proposed a combinatorial lower bound algorithm called in what follows CBA. This lower bound computes a set of *overflow items* iteratively in an optimal fractional packing by the following way; the authors define R to be the overall residual capacity of all the bins already initialized, whenever an item j is found, whose size is greater than R, a new bin is initialized and the overflow item of this bin is selected as the largest item among those already packed but not yet used as overflow items, then insert item j into the set O of overflow items in order to yield the maximum residual capacity for the next iteration. So this lower bound can be expressed by:

$$L_{CLB} = |O| \tag{3}$$

B. Upper bounds

Well known algorithms for the classical BPP: First Fit Decreasing(FFD) is one of the well-known algorithm often used to solve the classical BPP. Developed by Coffman et al. [3], FFD guarantees asymptotic worst case performance bounds of $11/9$ [6]. Let all items are sorted in the non-increasing weight order, the FFD algorithm consist on packing the current item in the first opened bin, else, a new bin is opened.

Other effective algorithm for solving the classical BPP called *Minimum Bin Slack (MBS)*. This algorithm was proposed by Gupta and Ho [7]. At each step, an attempt is made to find a set of items that fits the bin capacity as much as possible. At each stage, a list of items not assigned yet is sorted in the decreasing order of their weights. Each time a packing is determined, the items involved are placed in a bin and removed from a set of unpacked items. Thereafter Fleszar et al. [8] proposed a variant of MBS called MBS', in which the item of maximum size is not unstuck. MBS has been used in several variants of bin packing problem, such as the bin packing problem with conflicts [9], and the variable sized bin packing problem [10].

Ongkunaruk's heuristic for the OEBPP: Ongkunaruk [5] proposed a modification of the well-known first fit decreasing algorithm for the OEBPP called *Modified First Fit Decreasing (MFFD)*, this modification consists of determining the overflow items firstly, then apply the first fit decreasing for the rest of items not yet packed.

In MFFD, the author computes the lower bound K defined above (equation (2)) and considered as overflow items, the $K-$first large items , then apply the first fit decreasing algorithm for the remaining items i not yet packed, where $i = k + 1$ to n. Finally, assign the overflow items into the bins using FFD algorithm.

III. Problem Formulation

In this section, we propose a modification of the integer linear program initially proposed by Cesili and Righini [1]. The authors studied a variant of the open bin packing problem called the Ordered Open-End Bin Packing Problem (OOEBPP) in which items to be packed are sorted in a given order. For this formulation, the authors used the binary variable y_i to indicate whether item i is the overflow item in its bin, and the binary variable x_{ij} to indicate whether item i is assigned to the bin in which the overflow item is the item j. Because of the constraints on the order of the items, there is only x_{ij} variables with $i < j$. Instead, for the general OEBPP (without order), we have therefore x_{ij} decision variables with $i \neq j$.

$$Minimize \sum_{i \in N} y_i \tag{4}$$

s.t.

$$y_i + \sum_{i \neq j} x_{ij} = 1 \ \forall i \in N \tag{5}$$

$$\sum_{i \neq j} w_i x_{ij} \leq (C-1) y_j \ \forall j \in N \tag{6}$$

$$x_{ij} \in 0,1 \tag{7}$$

$$y_i \in 0,1 \tag{8}$$

Each binary variable y_i indicates whether item i is the overflow item in its bin. Each binary variable x_{ij} indicates whether item i is assigned to the bin in which the overflow item is item j. Constraints (5) impose that each item must be assigned to a bin, while constraints (6) impose that the overall weight of the items assigned to a bin, excluding the overflow item, must fit into the bin and must leave at least one capacity unit available for accommodating the overflow item. The number of bins used is indicated in the objective function (4) by the number of binary variables y_i set to 1.

IV. Proposed Heuristics

A. Adapted First Fit Decreasing AFFD

In this section, we will describe our heuristic algorithm, called in the following *Adapted First Fit Decreasing(AFFD).*

1) Main idea: In the open-end bin packing problem the capacity of any bin can be exceeded by only the last packed item, called the *overflow item*. Then, it is wise to consider the weightest items as overflow items in order to minimize the number of used bins. In this way, Ongkunaruk [5] proposed MFFD algorithm which is a modification of the FFD runway. The author determines overflow items firstly before packing, then apply first fit decreasing with bins capacity equal to $C-1$.

Our Proposition is an adaptation of the well-known first fit decreasing where each selected overflow item can occupy at least one unit of bin capacity. Sorted items in decreasing order of their weight, the AFFD algorithm consist on assignment of the current item i into the lowest indexed opened bin which accommodate it, else, a new bin is opened. An opened bin can receive an item i if and only if the residual capacity -without take into account the overflow item- is sufficient to contain it. In other words, an item i can be placed into an opened bin if the weight w_i is less than or equal to $C - 1 - \sum_{j\in\widehat{I}} w_j$ where $\widehat{I}$ is the set of items assigned to the opened bin, except the overflow item.

When a new bin is opened, a selection of an overflow item j is to be made. The overflow item j is selected as the weighted item among those already packed into all previous opened bins, but not yet used as overflow items. Once the overflow item is selected, a permutation of the placement between the selected item j and the current item i is performed.

2) AFFD Pseudo code: Let $\widehat{I}_j$ a set of items assigned to bin j, except the overflow item o_j. The AFFD heuristic can be summarized in the following steps:

- Step 0: Sort the items into a list I in decreasing order of their weight, set $i = 1$, $S = \varnothing$, $O = \varnothing$.
- Step 1: Until there are no items in a list I do
  - $S = S \cup i$;
  - Pack the current item i into the lowest indexed opened bin j^* in which the total weight of items $\widehat{I}_{j^*}$ already assigned to it *-except, the over flow item-* is strictly less than the bin capacity. Open a new bin if it cannot be assigned to any existing bin;
  - if a bin k is opened,
    - select the overflow item o_k such that $w_{o_k} = argmax_{i\in\widehat{I}}\{w_i\}$ we pack firstly the overflow item;
    - Permute placement of the current item i and selected overflow item o_k
    - Set $O = O \cup o_k$ and update $\widehat{I}$.
  - Set $I = I \setminus i$
  - Update the residual capacity of opened bins (We assume that the residual capacity of bin is the remaining of the capacity where we consider that the over flow item require only one unit)

3) Algorithm: Algorithm 1 details the AFFD procedure.

```
Algorithm 1: Adapted First Fit Decreasing AFFD
Data: $I$ Set of items ordered in decreasing order of
      weights, $w_i$ weight of item $i$, $C$ Bins capacity.
Result: The number of used bins $|O|$
initialization
$O := \emptyset$
$S := \emptyset$
for $i = 1$ to $n$ do
    $S = S \cup \{i\}$
    if ( i can be assigned in one of opened bins ) then
        assign item $i$ with bin capacity $C - 1$ using
         FFD procedure
    else
        $j = argmax_{j\in S}\{S\}$
        $S = S \setminus \{j\}$
        $O = O \cup \{j\}$
        Permute assignment between items $i$ and $j$
```

AFFD procedure can be implemented on $O(n \log n)$ time, where n is the number of concerning items.

B. Adapted Minimum Bin Slack AMBS

We describe in the follow the second proposed heuristic which consists in a simple adaptation of MBS heuristic of Gupta and Ho [7]. We call this proposition, Adapted Minimum Bin Slack heuristic (AMBS). In this heuristic, we execute the MBS procedure, in which we consider that the weight of the largest item in the current bin is equal to one capacity unit. Otherwise, this item is considered as the overflow item of its bin, and this bin must leave at least one capacity unit available for the requirement of the overflow item. The AMBS heuristic is summarized in the following:

Until there are no items in the unpacked items set I:

- Select the largest item in I and assign this item as an overflow item in the current bin j. The residual capacity of bin j becomes $r_j = r_j - 1$. Then, remove the selected overflow item from the set I;

- Apply the MBS-one search procedure on I with a capacity of bin equal to $C-1$, in which we obtain the best subset of unpacked yet items.
- Remove the founded subset from I.

The complexity of MBS procedure is $O(2^n)$, but the computational experience shows that it is quite efficient in solving practical problem instances with various parameters [8][9]. This is due to the fact that in practical problem instances, the number of items involved in each packing is much smaller than n. If the maximum number of items that can be placed in one bin is u, then the complexity of MBS-one-packing procedure is reduced to $O(n^u)$ and thus packings for all bins are built in $O(n^{u+1})$.

V. Computational Results

Heuristics and lower bounds have been coded in java and run on an Intel i5 @ 2Go of RAM. The solver CPLEX 12.5 was used to solve the linear programming to the optimality.

We tested our heuristics algorithm on three data-sets called *uniform*, *triplet* and *random* instances. The two first data-sets contain eight instances classes were initially proposed by Falkenauer [11] and they were excessively used to test the performances of several bin packing problem variants. Each class contains 10 different instances. The uniform data-set includes four instances classes of bins capacities $C = 150$ and items with integer weights uniformly distributed in the interval of $[20 - 100]$. The number of items n for each class is respectively, 120,250,500 and 1000. The second data-set was called the triplet bin-packing instances because in the classical bin-packing problem, each bin can be filled with at most three items. This data-set includes also four instances classes of bin capacities $C = 1000$ and items with integer weights uniformly distributed over the interval $[25 - 50]$ with one decimal digit. The number of items n for each class is respectively, 60, 120, 249 and 501.
If for the triplet instances in the classical bin-packing problem (closed bin), each bin cannot contains more than three items, so for the open bin cases we can add no more than one item as an overflow item. Therefore, a valid lower bound (lower number of used bins) can be expressed by $L^T_{OBP} = \lceil \frac{n}{4} \rceil$ where n is the problem size or the number of items.

The last data-set was generated randomly and contains five classes of instances with different problem size, $n = 50, 100, 200, 500$ and 1000. For each class, items weight was generated using uniform distribution over four different intervals $[20 - 140]$, $[20 - 160]$, $[40 - 140]$ and $[40 - 160]$. The capacity of bins is fixed to 200. Ten different instances was generated for a given problem size and weight interval distribution.

The performance and average run time of tested heuristics on data-sets described above are shown in Table I, Table II and Table III respectively. Each line contains average results over ten OEBPP instances and the best results are shown in bold faced characters.

Table headings are as follows:

Dev. :	Deviation of solution obtained by the corresponding heuristic(H) from the best lower bound provided by Ongkunaruk [5] LB^0_{OBP} and the lower bound provided by Cesili and Righini [1] LB_{CBA}, computed as :
sec. :	Computational time in seconds.

$$Dev(\%) = 100 * \frac{H - \max(LB^0_{OBP}, LB_{CBA})}{\max(LB^0_{OBP}, LB_{CBA})}$$

TABLE I. Computational Results for the Uniform Instances

Problem	MFFD		AFFD		AMBS	
size	*Dev.*	*sec.*	*Dev.*	*sec.*	*Dev.*	*sec.*
$u120$	**2.99**	*0*	**2.99**	*0*	3.65	*0.0031*
$u250$	1.41	*0.0085*	**0.94**	*0.0063*	2.50	*0.0031*
$u500$	1.33	*0.0266*	**1.02**	*0.014*	2.27	*0.0078*
$u1000$	1.42	*0.0312*	**0.95**	*0.0078*	2.10	*0.0328*
Average	1.78	*0.016*	**1.47**	*0.0070*	2.63	*0.0117*

TABLE II. Computational Results for the Triplet Instances

Problem	MFFD		AFFD		AMBS	
size	*Dev.*	*sec.*	*Dev.*	*sec.*	*Dev.*	*sec.*
$t60$	4.66	*0*	2.01	*0*	**0**	*0.0032*
$t120$	2.66	*0*	2.33	*0*	**0**	*0.0249*
$t249$	1.26	*0.0030*	1.26	*0*	**0**	*0.3557*
$t501$	1.34	*0.0157*	1.19	*0.0030*	**0**	*1.6488*
Average	2.48	*0.0047*	1.69	*0.0007*	**0**	*0.5081*

Table I shows computational results for the Falkenauer uniform instances, from these results we note that the average deviation decreases when the problem size increases, this is due to the convergence of the upper bounds and lower bounds from the optimal solution. In average, our heuristic AFFD performs better than heuristic provided by Ongkunaruk, more particularly when the problem size increase, this is due to the way on which MFFD algorithm extract the set of overflow items. In fact, the lower bound provided by Ongkunaruk performs worse when the problem size increase. Ongkunaruk's upper bound MFFD gives a further solution.

Concerning AMBS, this heuristic gives an important deviation from the optimal solution, this is due to the fact that from the MBS adaptation, some items with a large weight will be packed inside the bin. While, the optimal solution requires that all largest weight items must be packed as *overflow items* in order to minimize the number of used bins.

In Table II, we present the computational results of MFFD and proposed heuristic for the Falkenauer triplet instances. From these results, the overall attracting remarks that our heuristic AMBS give the optimal solution for this instances, because it is enough to find three items in each subset to be packed inside the bin, and one as *overflow item*. So the solution given by AMBS coincides with the lower bound value $L^T_{OBP} = \lceil \frac{n}{4} \rceil$. Moreover, from this table we note that both proposed heuristics AMBS and AFFD perform better than MFFD.

For a given problem size, the results given by both AFFD and MFFD are better on the uniform instances comparing to the triplet instances. These remarks can be explaining by the fact that the Falkenauer triplet instances are particularly difficult because in the optimal solution all constructed bins should have maximally three items as no overflow item plus one overflow item, therefore it is difficult to find an optimal subset of four items to each bin using first fit decreasing.

In overall, all the results present reasonable average computation time which confirms the good performances of our proposed heuristics. The time required by MFFD and AMBS are negligible. Furthermore, it is obvious to note that the time requirements increase with the problem size, but reasonable in order of some millisecond until problem size of 500 items.

Table III shows computational results for the random instances. In addition to both average deviation and execution time columns, we show the column $Diff$ which represent the average difference of number of bins obtained by the proposed heuristics and the lower bound. From these results we remark that the average deviation decreases when the problem size increases, this is due to the convergence of the upper bounds and lower bounds from the optimal solution. The results also show that MFFD and AFFD methods performs better in quality of solution and computational time. An average deviation of 0% to 2.58% is given for the small problem size and 0.47% to 2.75% for the problems beyond to 200 items. Also, for such an interval of weight distribution, the average deviations decreases when the problem size increases which ensures the good performances of proposed methods. In overall proposed AFFD performs better with an overall average deviation of 1.29%. However, *MFFD* and *AMBS* show good performances and the same behaviour in the variation of deviation with an overall average deviation of 1.36% and 6.33% respectively. Although we have positive values of deviation, the average difference between the lower bound and the upper bounds is usually just two bins for MFFD and AFFD, while this difference is in order to average seven bins for AMBS. Obviously, this value of the difference increases with the problem size.

Generally, obtained results are given with reasonable average computation time, in order to some milliseconds for MFFD and AFFD heuristics and few large running time for AMBS, but always in order to milliseconds. The execution time increase with the problem size. For 1000 items, the solutions are obtained after average 28, 10 and 149 milliseconds for MFFD, AFFD and AMBS respectively. These results confirm the excellent performances of proposed heuristics.

VI. Conclusion

The open end bin-packing problem is an NP-hard combinatorial problem often encountered in the practical field. Few works are carried out to solve the problem in polynomial or pseudo-polynomial time. Through the present paper, we have proposed and described two newly heuristics for the OEBPP problem, these heuristics are an adaptation of the well-known first fit decreasing algorithm and minimum bin slack algorithm.

Computational results based on a benchmark test bed show the good performance of proposed heuristics both on quality of solution, and on required execution time.

Acknowledgement

This work was supported by Algerian state and performed jointly at LMA/EMP (Laboratoire de mathématiques appliquées de l'Ecole Militaire Polytechnique - Alger, Algérie) and CRIL (Centre de Recherche en Informatique de Lens CNRS - UMR 8188, France). This support is gratefully acknowledged.

References

[1] A. Ceselli and G. Righini, "An optimization algorithm for the ordered open-end bin-packing problem," *Operations Research*, vol. 56, no. 2, pp. 425–436, 2008.

[2] J. Y. T. Leung, M. Dror, and G. H. Young, "A note on an open-end bin packing problem," *Journal of Scheduling*, vol. 4, no. 4, pp. 201–207, 2001.

[3] J. E. G. Coffmann, M. R. Garey, and D. S. Johnson, *Approximation algorithms for NP-hard problems: Approximation algorithms for bin-packing-a survey*. Boston: PWS Publishing, 1997, ch. 2, pp. 46–96.

[4] J. Yang and J. Y. T. Leung, "The ordered open-end bin-packing problem," *Operations Research*, vol. 51, no. 5, pp. 759–770, 2003.

[5] P. Ongkunaruk, "Asymptotic worst-case analyses for the open bin packing problem," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2005.

[6] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bound for simple one dimensional packing algorithms," *SIAM Journal on computing*, vol. 3, no. 4, pp. 299–325, 1974.

[7] J. N. D. Gupta and J. C. Ho, "A new heuristic algorithm for the one-dimensional bin-packing problem," *Production Planning & Control*, vol. 10, no. 6, pp. 598–603, 1999.

[8] K. Fleszar and K. S. Hindi, "New heuristics for one-dimensional bin-packing," *Computers & Operations Research*, vol. 29, pp. 821–839, 2002.

[9] M. Maiza and M. S. Radjef, "Heuristics for solving the bin-packing problem with conflicts," *Applied Mathematical Sciences*, vol. 5, no. 35, pp. 1739 – 1752, 2011.

[10] M. Maiza, A. Labed, and M. S. Radjef, "Efficient algorithms for the offline variable sized bin-packing problem," *Journal of Global Optimization*, vol. 57, no. 3, pp. 1025–1038, 2013.

[11] E. Falkenauer, "A hybrid grouping genetic algorithm," *Journal of heuristics*, vol. 2, pp. 5–30, 1996.

TABLE III. COMPUTATIONAL RESULTS FOR THE RANDOM INSTANCES

		MFFD			AFFD			AMBS		
		% Dev	*Diff*	*Sec.*	*% Dev*	*Diff*	*Sec.*	*% Dev*	*Diff*	*Sec.*
R-50	[20-140]	1.66	0.2	*0.0000*	**0.83**	0.1	*0.0000*	3.21	0.4	*0.0000*
	[20-160]	**0.00**	0.0	*0.0000*	**0.00**	0.0	*0.0000*	5.07	0.7	*0.0000*
	[40-140]	**1.42**	0.2	*0.0000*	**1.42**	0.2	*0.0000*	2.14	0.3	*0.0000*
	[40-160]	**1.91**	0.3	*0.0000*	2.58	0.4	*0.0000*	3.83	0.6	*0.0000*
	Avg.	1.24	0.2	*0.0000*	**1.20**	0.2	*0.0000*	3.56	0.4	*0.0000*
R-100	[20-140]	**1.16**	0.3	*0.0016*	**1.16**	0.3	*0.0000*	5.84	1.5	*0.0000*
	[20-160]	**0.35**	0.1	*0.0000*	**0.35**	0.1	*0.0000*	5.09	1.4	*0.0000*
	[40-140]	2.50	0.7	*0.0000*	**2.15**	0.6	*0.0015*	5.36	1.5	*0.0000*
	[40-160]	**2.07**	0.6	*0.0016*	2.40	0.7	*0.0000*	5.79	1.7	*0.0000*
	Avg.	1.52	0.4	*0.0008*	**1.51**	0.4	*0.0003*	5.52	1.5	*0.0000*
R-200	[20-140]	**0.79**	0.4	*0.0000*	1.18	0.6	*0.0000*	6.94	3.5	*0.0000*
	[20-160]	**0.75**	0.4	*0.0000*	0.94	0.5	*0.0000*	7.72	4.1	*0.0000*
	[40-140]	2.34	1.3	*0.0016*	**1.44**	0.8	*0.0000*	6.29	3.5	*0.0015*
	[40-160]	**1.71**	1.0	*0.0000*	**1.71**	1.0	*0.0016*	7.00	4.1	*0.0016*
	Avg.	1.39	0.8	*0.0004*	**1.31**	0.7	*0.0004*	6.98	3.8	*0.0007*
R-500	[20-140]	**0.47**	0.6	*0.0046*	0.72	0.9	*0.0031*	8.14	10.1	*0.0031*
	[20-160]	**0.82**	1.1	*0.0000*	0.89	1.2	*0.0000*	7.85	10.5	*0.0016*
	[40-140]	2.68	3.7	*0.0061*	**1.74**	2.4	*0.0032*	6.97	9.6	*0.0093*
	[40-160]	**1.56**	2.3	*0.0031*	1.97	2.9	*0.0047*	7.55	11.1	*0.0032*
	Avg.	1.38	1.9	*0.0034*	**1.33**	1.9	*0.0027*	7.62	10.3	*0.0043*
R-1000	[20-140]	**0.73**	1.8	*0.0265*	0.77	1.9	*0.0110*	8.65	21.3	*0.2511*
	[20-160]	**0.53**	1.4	*0.0173*	0.68	1.8	*0.0094*	8.61	22.6	*0.0764*
	[40-140]	2.75	7.6	*0.0407*	**1.48**	4.1	*0.0111*	7.35	20.3	*0.2683*
	[40-160]	**1.22**	3.6	*0.0264*	1.63	4.8	*0.0092*	7.39	21.7	*0.0014*
	Avg.	1.30	3.6	*0.0277*	**1.14**	3.1	*0.0101*	8.00	21.5	*0.1493*
Total Avg.		1.36	1.4	*0.0064*	**1.29**	1.3	*0.0027*	6.33	7.5	*0.0308*