Three Algorithms for Packing Variable Size Bobbins

Uwe Schöning Jacobo Toran Thomas Thierauf

Jochen Messner Uwe Bubeck

Abt. Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

1 Introduction

The problem we deal with consists in finding efficient ways for packing fiber-bobbins on pallets, optimizing the space used. The packing takes place in a factory environment. The bobbins are stored in a room in the factory and are brought to the packing area on multi-trays. 21 bobbins fit in each multi-tray. The pallets are standard rectangles of size 100×120cm. The bobbins have all the same height but their diameter is variable and can range from 9 to 29cm. A mobile robot packs the bobbins on the pallets and it is assumed that the robot can reach the bobbins on the multi-tray in any given order.

For a given placement of bobbins on the pallet, the *occupancy* is the percentage of the pallet occupied by the bobbins. The problem of placing bobbins optimally can be formalized in the following way:

> Given a rectangle of a fixed size (in our case $100\times$ 120 cm), and a set of circles of variable radius, find a non-overlapping placement of the circles in the rectangle maximizing the occupancy.

This is a very hard problem in mathematics. A special case of this problem that has been studied for many years is the simpler situation in which all the circles have the same radius. Even in this case, and for a relatively small amount of circles the problem is still unsolved. More precisely, it is not known what is the size of the smallest square where 21 circles of radius 1 can fit. Because of this fact we do not attempt to obtain an optimal solution for the case of circles with variable radius but rather give approximate solutions that can work well in practice.

We have developed the following three algorithms that work according to different restrictions provided by the actual implementation in the factory:

1. General Algorithm,

2. Layer Algorithm and

3. Buffer Algorithm.

The first algorithm is the most general one and provides the base for the other two. It needs to know in advance the information about the sizes of all the bobbins that are to be placed in one pallet, before placing the first bobbin. This algorithm achieves the best packing performance, but as discussed later, it presupposes conditions on the robot that might be unrealistic.

The Layer Algorithm only needs the information of the sizes of the bobbins coming in the next multi-tray (21 bobbins). It packs the bobbins in independent stages, placing on the pallet the bobbins of one multi-tray before considering the ones on the next multi-tray. Under these realistic conditions, the average occupancy obtained by version-b of this algorithm is a little smaller than the one from the General Algorithm.

Finally, the Buffer Algorithm works like the Layer Algorithm (only the sizes of the 21 bobbins in the next multi-tray to be packed are known in advance) but considers that there is an intermediate buffer in the packing zone where up to k bobbins can be temporarily stored. We will see in the test section occupancy results for different small values of k. The case $k = 21$ is equivalent to considering that at some moments of the packing two full multi-trays (with 21 bobbins each) are present.

2 Description of the Algorithms

2.1 The General Algorithm

The algorithm supposes that the information about the sizes of a large number of bobbins that potentially fit on the current pallet is known. Considering that 9cm is the smallest bobbin diameter and that the pallets are 100×120cm rectangles, these can mean information about up to 160 bobbins that has to be known before the position of any of them can be computed. A further restriction of the algorithm is that the final position of each one of the bobbins can be practically anywhere on the pallet. When the bobbins arrive on the multi-tray they are placed at their assigned position. A situation that can happen is that at some packing stage, "holes" to be filled by bobbins arriving later in the process can be left somewhere in the middle of the pallet (see Figure 1). The robot should be able to place then these bobbins in the "holes" when they arrive. This might be unrealistic depending on the possible movements of the robot.

In spite of these facts we include this algorithm since it provides the base for the other two and moreover, its performance with bobbins of different sizes provides a good benchmark to compare the occupancy achieved by the more realistic Algorithms 2 and 3.

The general idea of the algorithm is to place virtually a set of circles on a rectangle that initially is much larger than the real pallet. On different refinement steps the circles are shifted by a small amount in the direction of the middle of the rectangle and when the rectangle sides do not touch any of the circles then the sides are shrinked and the rectangle becomes smaller. The refinement steps are repeated until either the rectangle fits the actual pallet size (in this case we have found a positioning for the circles), or after several trials the algorithm considers that it is not possible to pack that many circles. In this second case the last circle is taken out of the rectangle and the algorithm starts from the beginning with one less circle.

Border contraction: The border contraction in this algorithm is a little tricky: suppose, after one "global move", the left border could be reduced by 1cm and the right border by 5cm. In this case, all circles are (virtually) moved to the right so that both borders are actually reduced by 3cm. A second point is that after each "global move" only the left-right or the upper-lower borders are reduced, but not both, depending on which is further away from the final pallet size. Also after the left-right or the upper-lower borders are within the pallet size, they "freeze" (to the pallet size), and only the other borders are tried to be reduced afterwards.

Observation: As mentioned before, a disadvantage of this algorithm is that the final position of the bobbins in the pallet might be very different than the arriving order of the multi-trays. However, as shown in Figure 1, if the position of the bobbins at the initial stage of the algorithm follows a given order, there are only a few bobbins whose final position is very different from the initial relative order. This fact gives hope for the practical use of the General Algorithm.

2.2 The Layer Algorithm

The second algorithm is an adaptation of the the general one to the situation in which only the size of the 21 bobbins on a multi-tray is known.

Initially 21 circles are virtually placed on a surface that is as wide as the pallet but open on the top. The circles move in small steps randomly towards the bottom of the pallet. Instead of shrinking a rectangle around the circles like in the General Algorithm, in this case a horizontal line on top of all the circles tries to move down as much as possible. When the line cannot move further down, the algorithm checks all the 21 circles in the order given by the y coordinate of the center and shifts them down if this is possible. This is done since it can be the case that the vertical line cannot

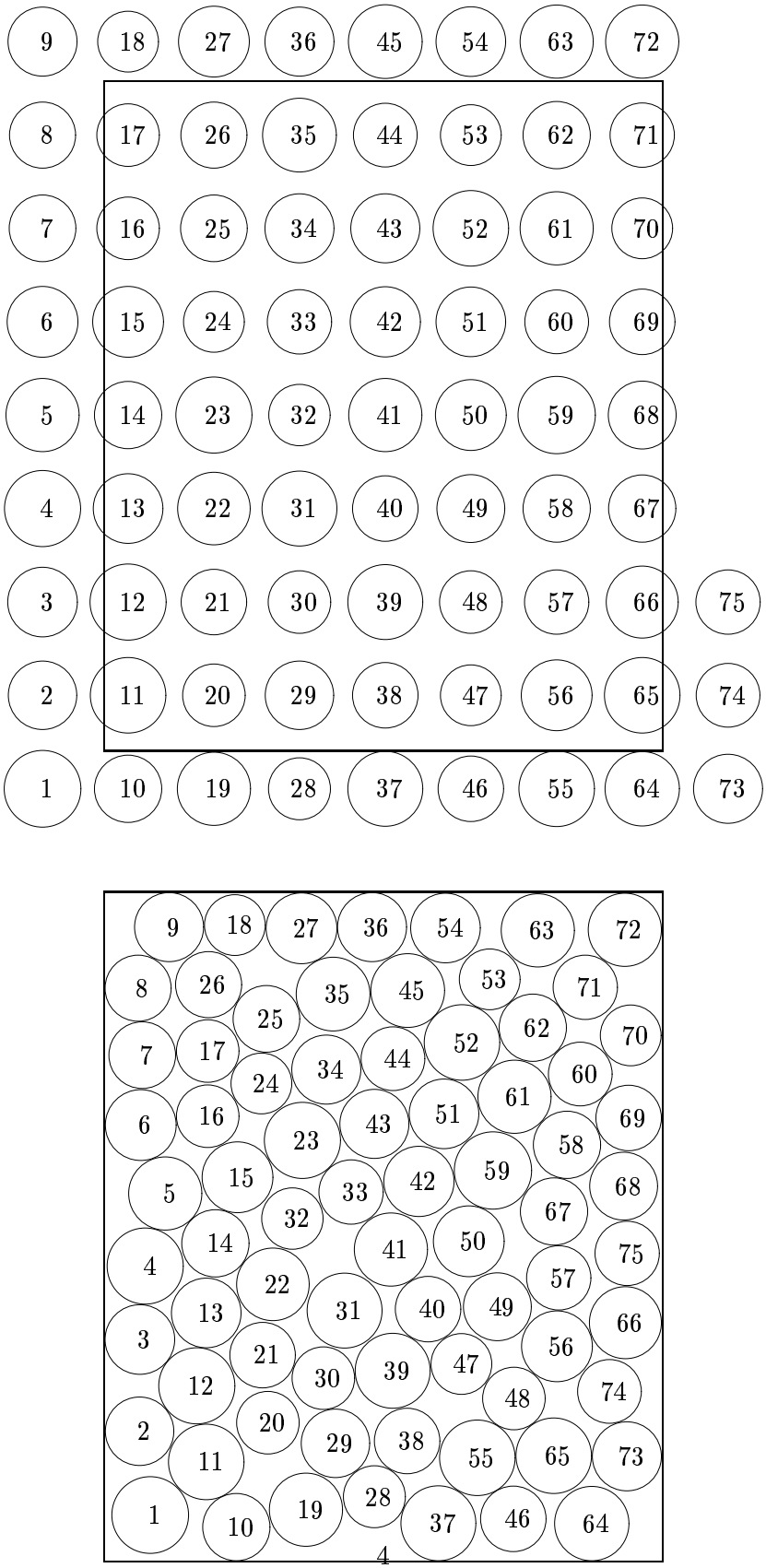

Figure 1: The initial positioning of the circles on a bigger rectangle at the beginning, and the final situation computed by the General Algorithm.

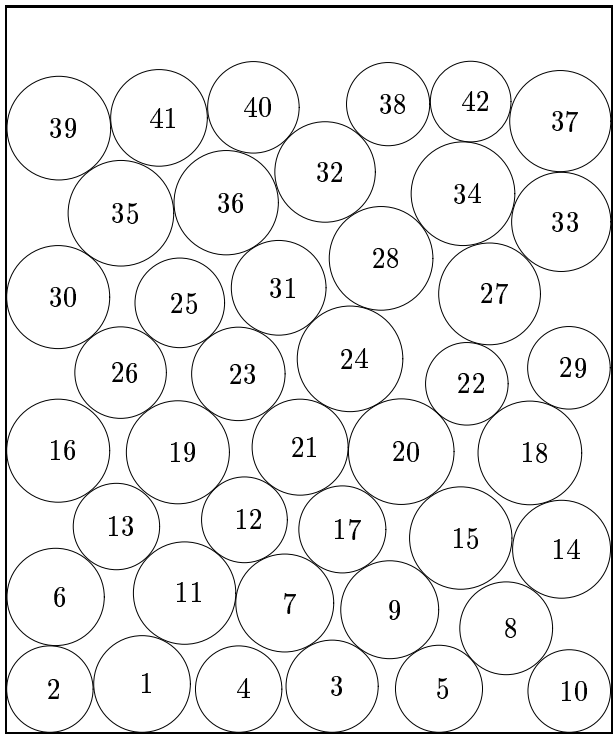

Figure 2: Version-a the layer Algorithm. After two stages there is some space left at the top. The occupancy is now 69.9 percent.

move further down but some of the circles are not optimally placed towards the bottom. In a subsequent filling stage this can waste space.

The procedure continues then with the next block of 21 circles in the same manner, and follows the same procedure until no more circles fit at the top. In its simplest version (version-a) the algorithm finishes at this stage.

A negative situation that can happen is in the case there is still some room left along the top of the pallet but this room is not "high" enough to fit one circle (Figure 2).

When this case is detected after virtually placing the last 21 circles, the algorithm takes them again out of the pallet (the bobbins were only virtually there) and places them in a different way trying to concentrate the empty space at a position at the middle. For doing so, instead of sliding a vertical line towards the bottom of the pallet, the algorithm pushes the bobbins from the top with a circle that is almost as wide as th pallet (Figure 3). As a result, the empty space is now at the top of the pallet but concentrated in the middle. The algorithm places then in the standard way new bobbins, coming from the next multi-tray, on the hole. It tries first to fit a certain

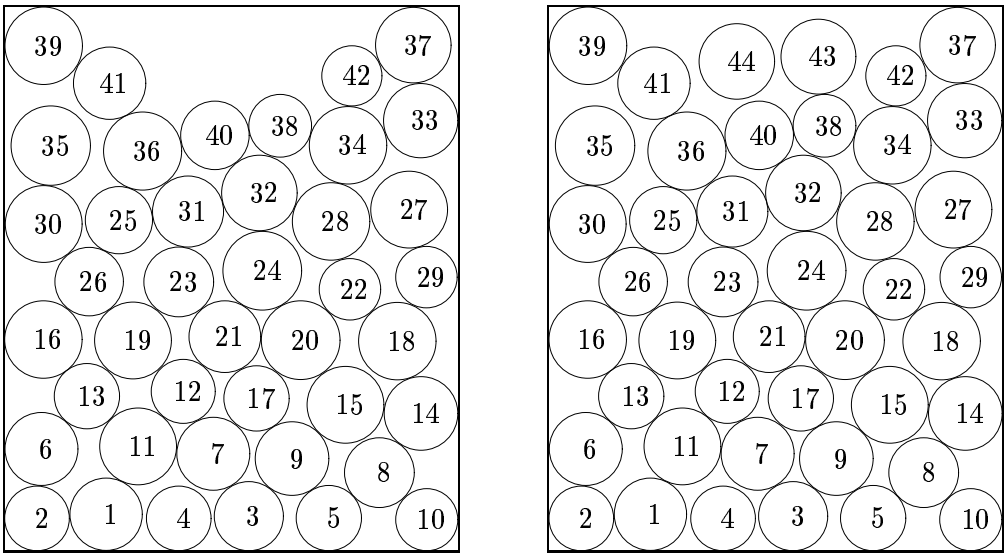

Figure 3: Version-*b* of the Layer Algorithm. The left figure shows the same example as in Figure 2 with the empty space concentrated at the middle. On the right figure the algorithm has placed two more circles in this space. The occupancy is improved to 73.6%.

number of bobbins and if they cannot all be fitted tries again with one less bobbin until they fit.

Since this algorithm works in layers, it has the nice property that the bobbins can be packed in order, from the bottom to the top of the pallet. The positions of the current 21 bobbins lie on top of the previous 21. The robot can consider any nice packing order for the 21 bobbins (for example from left to right and from bottom to top) and then take them from the multi-tray according to this order by considering the numbering given by the computed solution.

2.3 The Buffer Algorithm

One of the differences between the General Algorithm and the one packing in layers, is that in the later the negative situation can arise in which some empty space is left at the top of the pallet (Figure 2). Algorithm 2b solved this situation by trying to concentrate the empty space in the middle. In Algorithm 3, we consider a different way of dealing with this problem by using an additional storage buffer where some of the bobbins can be temporarily stored. For explanation purposes we consider that the buffer has a certain size k. We have tested the algorithm for the values of $k = 10$ and $k = 21$. This last case is equivalent to the situation in which at the end of

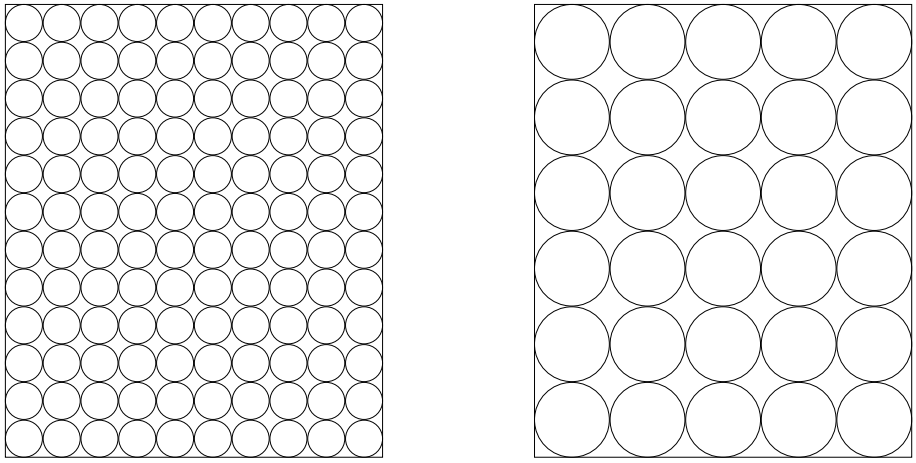

Figure 4: The square grid with 10 and 20cm diameter circles. Occupancy=78,53 percent

the packing process of a pallet, two multi-trays are present.

The algorithm works like Algorithm 2 but when arriving at a negative final situation in which an empty margin is left at the top, the algorithm takes the last block of 21 bobbins out of the pallet (the bobbins were only virtually there), places k of them on the buffer and puts the rest $(21 - k)$ on the pallet following the same procedure as before. The algorithm waits then for the next multi-tray with 21 bobbins. It then places on the pallet as many bobbins as possible, taking them from the buffer and the multi-tray $(21 + k)$. There are now more bobbins available as they can fit. The ones that do not fit are placed at the beginning of the next pallet to be filled and the process continues.

The advantage of this temporary storage on the buffer is that at the final stage when the last set of bobbins has to be placed, the empty margin at the top of the pallet is bigger than before (there are k bobbins less on the pallet). The algorithm can then try to fit a larger number of bobbins (more similar to the General Algorithm) and the occupancy for the last part of the pallet might be improved.

As shown by the tests, the third algorithm is better than the a-version of the Layer Algorithm but compared to the b version it only makes some (small) occupancy improvement in the cases with small diameter circles and it is worse otherwise.

3 Considerations on the Maximum Occupancy

In order to be able to evaluate the performance of our algorithms we can compare the occupancy obtained by our solutions with the one achieved by

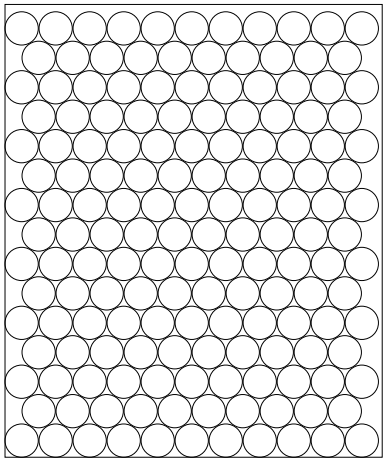

Figure 5: Hexagonal packing with 9cm diameter circles. Occupancy=83.72 percent.

two kinds of regular packings that work well for the case of circles of the same radius: the square and hexagonal grids.

The **square grid** is obtained by placing the circles "on top of each other" as shown in Figure 4. The best occupancy for this packing is obtained when the diameter of the circles divides the length of both rectangle sides. If this condition holds, the occupancy achieved is approximately 78,53 percent independently of the circle size. Without this condition the occupancy can be much worse, for example for 22cm diameter circles the occupancy achieved is about 63.35%.

The **hexagonal grid** shown in Figure 5 is obtained by placing a first row of circles at the bottom of the rectangle starting at the left side and then placing further rows on top shifted in one direction by the radius size. For the range of diameters considered here (between 9 and 28cm), the maximum occupancy is achieved with 9cm diameter circles. In this case the occupancy is approximately 83.76 percent. Again, for other circle size the occupancy is worse. For example, for 22cm diameter circles, the hexagonal grid achieves an occupancy of about 76.02%.

These two examples of packing strategies can only be uses in case of equal size circles, but give an indication to estimate the quality of our algorithms for variable size circles.

4 Conclusions

We have developed three algorithms for the problem of packing variable size bobbins. The General Algorithm achieves very good occupancy results, on average over 75% for bobbins of up to 20cm diameter, and over 70% when

bigger bobbins can arrive. This algorithm however needs to "reserve" in advance all the bobbins to be packed and moreover, the optimal order in which the bobbins should be packed by the robot might not correspond with the order in which the bobbins arrive on the multi-trays.

The Layer and Buffer algorithms are more realistic since in both cases the bobbins can be placed on the pallet in any desired order (for example left to right and bottom to top). The Layer Algorithm in its b variant achieves on average an occupancy of about 3% less that the General Algorithm, which we consider to be a good performance.

The Buffer Algorithm can store temporarily some of the bobbins in an extra buffer. For a buffer of size 10, only in a very few cases is the occupancy achieved by this algorithm better than the one from the b version of the Layer Algorithm and therefore we consider this buffer not to be really necessary. For a buffer of size 21, the algorithm presents an occupancy improvement over the b version of the Layer Algorithm of about 1% for the case of small diameter circles (up to a diameter of about 19cm). For larger circles the occupancy is not better than that of the Layer Algorithm.

All the algorithms work very fast. The execution times shown with the tests are measured on a small personal computer with a Pentium III and 600 MHz processor. A single execution of one of the algorithms takes only a few seconds (less than 10).

It can be seen from our results that the algorithms work better when the bobbins have relative small diameter (up to approx. 20cm). In case there are bobbins of larger size, it does not matter if they come mixed with small bobbins. On the contrary, the occupancy results become better in this case.

5 Occupancy Tests

We present the occupancy results for the different algorithms obtained by computer simulations. We have done for each one of the algorithms (General Algorithm, versions a and b of the Layer Algorithm, and Buffer Algorithm with a buffer of size 10 and 20) and for each interval for the possible diameter of the bobbins, **25** simulations. In the simulations the bobbins sizes were generated at random under a uniform distribution of the possible values in the interval. The intervals are divided in three groups: the small ones (a difference of 1 and 1.5cm), medium (a difference of 3cm) and large intervals (difference larger that 3cm). We give the data for the worst, best and average occupancy results for the 25 simulation. In parenthesis we include the number of bobbins packed in each one of the cases.

Small Intervals

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
9–10	General	77.7	(130)	78.3	(134)	**78**	(132)	3s
	Layer a	73.9	(125)	75.4	(127)	**74.5**	(125)	4s
	Layer b	74.2	(126)	76.8	(130)	**75.6**	(128)	6s
	Buffer 10	73.8	(125)	76.1	(129)	**75**	(127)	8s
	Buffer 21	73.9	(126)	75	(127)	**74.6**	(132)	2s
10–11	General	77.7	(107)	78.4	(110)	**78**	(108)	3s
	Layer a	72.5	(101)	75.7	(105)	**74.4**	(103)	5s
	Layer b	72.4	(101)	75.5	(105)	**74.4**	(103)	4s
	Buffer 10	69.3	(95)	75.7	(105)	**72.6**	(100)	6s
	Buffer 21	74.2	(103)	76.3	(105)	**75.4**	(104)	2s
11–12	General	77.2	(89)	78.5	(91)	**77.9**	(90)	3s
	Layer a	71.8	(84)	74	(86)	**72.9**	(84)	3s
	Layer b	73	(84)	76.2	(88)	**74.6**	(86)	4s
	Buffer 10	71.8	(83)	75.1	(87)	**73.6**	(85)	6s
	Buffer 21	71.9	(84)	74.2	(86)	**72.9**	(84)	28s
12–13	General	76.6	(75)	78.8	(77)	**77.9**	(76)	3s
	Layer a	72.5	(71)	79.1	(78)	**76.5**	(74)	4s
	Layer b	73.7	(72)	77	(75)	**75.7**	(74)	5s
	Buffer 10	73.4	(72)	78.1	(77)	**75.8**	(74)	6s
	Buffer 21	75.6	(74)	79	(77)	**77.7**	(75)	3s
13–14	General	76.7	(64)	78.8	(66)	**77.9**	(65)	2s
	Layer a	72.9	(61)	75.5	(63)	**74.9**	(62)	3s
	Layer b	72.6	(61)	77.7	(66)	**75.2**	(63)	4s
	Buffer 10	71.3	(60)	75.9	(64)	**73**	(61)	4s
	Buffer 21	74.3	(63)	76.6	(65)	**75.2**	(63)	1s
14–15	General	75.1	(55)	78.1	(57)	**76.4**	(55)	3s
	Layer a	70.6	(51)	75.8	(55)	**74**	(53)	4s
	Layer b	71.7	(52)	75.7	(55)	**73.1**	(53)	4s
	Buffer 10	71.4	(52)	76.6	(56)	**73.7**	(53)	5s
	Buffer 21	72.9	(53)	76.3	(55)	**74.7**	(54)	1s
15–16	General	75.9	(48)	78.4	(50)	**77.2**	(49)	2s
	Layer a	67.7	(43)	76.1	(48)	**72.9**	(46)	2s
	Layer b	71.5	(45)	76.1	(48)	**73.4**	(46)	3s
	Buffer 10	71.7	(46)	77.1	(49)	**74.2**	(47)	4s
	Buffer 21	73.4	(47)	76.3	(49)	**74.7**	(47)	1s
16–17	General	73.8	(41)	77	(43)	**75.8**	(42)	1s
	Layer a	70.9	(40)	75.3	(42)	**74.5**	(41)	2s
	Layer b	72.1	(40)	76.8	(43)	**74.2**	(41)	2s

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
	Buffer 10	69.6	(39)	78.8	(44)	**75.9**	(42)	2s
	Buffer 21	73.1	(41)	77.1	(43)	**75.8**	(42)	1s
17–18	General	74	(37)	76.9	(38)	**75.5**	(37)	1s
	Layer a	71.5	(36)	76.3	(38)	**74.3**	(37)	1s
	Layer b	69.6	(35)	75.5	(38)	**73.6**	(36)	2s
	Buffer 10	68.4	(34)	76.2	(38)	**73.8**	(36)	2s
	Buffer 21	73	(36)	76.1	(38)	**74.6**	(37)	3s
18–19	General	73.9	(33)	77.6	(35)	**75.6**	(33)	1s
	Layer a	67.4	(30)	74.4	(33)	**72**	(32)	1s
	Layer b	69.2	(31)	74.2	(33)	**71.6**	(31)	2s
	Buffer 10	67.6	(30)	72.2	(32)	**71.4**	(31)	2s
	Buffer 21	71.4	(32)	76.3	(34)	**73.7**	(32)	3s
19–20	General	69.9	(28)	77.4	(31)	**73.1**	(29)	1s
	Layer a	70.1	(28)	75.1	(30)	**72.8**	(29)	2s
	Layer b	70	(28)	74.4	(30)	**72.1**	(28)	2s
	Buffer 10	69.5	(28)	74.8	(30)	**72.9**	(29)	2s
	Buffer 21	71.3	(29)	74.9	(30)	**72.6**	(29)	2s
20–21	General	71.4	(26)	76.5	(28)	**73.9**	(26)	1s
	Layer a	70.5	(26)	72	(26)	**71.5**	(26)	1s
	Layer b	69.2	(25)	74.4	(27)	**71.7**	(26)	2s
	Buffer 10	68.9	(25)	72.1	(26)	**71.1**	(25)	2s
	Buffer 21	70.7	(26)	74.6	(27)	**72.2**	(26)	2s
21–22	General	74.8	(25)	76.3	(25)	**75.5**	(25)	1s
	Layer a	63.8	(21)	72.5	(24)	**68.1**	(22)	1s
	Layer b	69.1	(23)	75.6	(25)	**72.3**	(23)	1s
	Buffer 10	67.2	(22)	72.9	(24)	**71.3**	(23)	1s
	Buffer 21	69.3	(23)	75.2	(25)	**72.2**	(23)	2s
22–23	General	71.8	(22)	77	(23)	**74.2**	(22)	1s
	Layer a	68.8	(21)	70.2	(21)	**69.5**	(21)	1s
	Layer b	68.5	(21)	73.5	(22)	**69.9**	(21)	1s
	Buffer 10	69.2	(21)	73.6	(22)	**70.9**	(21)	1s
	Buffer 21	69.5	(21)	73.5	(22)	**72.1**	(21)	2s
23–24	General	68.2	(19)	72.4	(20)	**70.3**	(19)	1s
	Layer a	68.2	(19)	72.6	(20)	**70.2**	(19)	1s
	Layer b	68.2	(19)	72.5	(20)	**70.7**	(19)	2s
	Buffer 10	68	(19)	71.8	(20)	**69.1**	(19)	2s
	Buffer 21	68.2	(19)	72.5	(20)	**70.3**	(19)	1s
	General	67.6	(17)	75	(19)	**71.2**	(18)	1s
	Layer a	69.9	(18)	74.9	(19)	**71.9**	(18)	1s

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
24–25	Layer b	70.1	(18)	75.2	(19)	**71.9**	(18)	2s
	Buffer 10	67	(17)	75.3	(19)	**72**	(18)	2s
	Buffer 21	67.2	(17)	75.3	(19)	**71.7**	(18)	1s
25–26	General	67.5	(16)	73	(17)	**71.4**	(16)	1s
	Layer a	67.6	(16)	72.9	(17)	**70.3**	(16)	1s
	Layer b	67.6	(16)	72.8	(17)	**70.8**	(16)	2s
	Buffer 10	59.1	(14)	72.9	(17)	**69.1**	(16)	2s
	Buffer 21	67.3	(16)	73.2	(17)	**70.7**	(16)	1s
26–27	General	73	(16)	74.3	(16)	**73.6**	(16)	1s
	Layer a	68.3	(15)	73.7	(16)	**70**	(15)	1s
	Layer b	68	(15)	73.5	(16)	**69.4**	(15)	2s
	Buffer 10	64.4	(14)	74	(16)	**69.4**	(15)	2s
	Buffer 21	68	(15)	74	(16)	**69.8**	(15)	1s
27–28	General	68.8	(14)	74.3	(15)	**71.2**	(14)	1s
	Layer a	64	(13)	69.8	(14)	**67.9**	(13)	1s
	Layer b	64.4	(13)	74.6	(15)	**69.6**	(14)	1s
	Buffer 10	64.1	(13)	69.9	(14)	**68.3**	(13)	1s
	Buffer 21	64.3	(13)	73.1	(15)	**68.1**	(13)	1s
28–29	General	68.4	(13)	69.6	(13)	**69.1**	(13)	1s
	Layer a	58.7	(11)	69.5	(13)	**67.3**	(12)	1s
	Layer b	63.5	(12)	69.8	(13)	**67.8**	(12)	1s
	Buffer 10	58	(11)	69.2	(13)	**61.3**	(11)	1s
	Buffer 21	68.2	(13)	69.9	(13)	**69.1**	(13)	1s
9–10.5	General	77.7	(124)	78.3	(127)	**78**	(125)	3s
	Layer a	72.2	(116)	77.3	(125)	**75.3**	(120)	4s
	Layer b	73.4	(118)	77.3	(125)	**75.6**	(121)	7s
	Buffer 10	72	(116)	75.8	(122)	**73.9**	(118)	8s
	Buffer 21	75.1	(120)	77.3	(125)	**76.3**	(122)	4s
10.5–12	General	77.2	(93)	78.5	(96)	**78**	(94)	3s
	Layer a	68.9	(84)	75.4	(91)	**71.4**	(85)	4s
	Layer b	72.8	(87)	75	(91)	**73.9**	(88)	5s
	Buffer 10	72.5	(87)	75.1	(91)	**73.8**	(89)	6s
	Buffer 21	70.2	(84)	77.1	(93)	**74.2**	(89)	2s
12–13.5	General	76.6	(71)	78.7	(75)	**77.6**	(72)	4s
	Layer a	74.6	(71)	76.9	(72)	**75.5**	(71)	4s
	Layer b	73.6	(69)	77.4	(73)	**75.4**	(70)	5s
	Buffer 10	73.5	(69)	76.8	(72)	**75.2**	(70)	7s
	Buffer 21	75.1	(70)	78.1	(73)	**76.6**	(71)	3s
	General	76.3	(57)	78.5	(59)	**77**	(57)	3s

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
13.5–15	Layer a	73	(55)	76	(57)	**74.3**	(56)	3s
	Layer b	72.2	(53)	75.4	(57)	**73.9**	(55)	4s
	Buffer 10	72.8	(54)	77.6	(58)	**74.8**	(56)	5s
	Buffer 21	72.7	(55)	76.7	(58)	**75**	(56)	1s
15–16.5	General	75.8	(46)	77.6	(48)	**76.6**	(47)	1s
	Layer a	67.7	(42)	72.6	(45)	**69.8**	(42)	1s
	Layer b	71.3	(44)	77.5	(48)	**73.3**	(45)	2s
	Buffer 10	71.4	(44)	75.4	(47)	**73.5**	(45)	3s
	Buffer 21	73.1	(45)	76	(47)	**74.1**	(45)	2s
16.5–18	General	73.6	(38)	77.4	(40)	**75.4**	(38)	1s
	Layer a	71.2	(37)	76.2	(39)	**73.4**	(37)	2s
	Layer b	71.5	(37)	74.9	(38)	**73.3**	(37)	2s
	Buffer 10	72.5	(37)	75.1	(38)	**73.8**	(37)	2s
	Buffer 21	72.2	(37)	77.5	(40)	**75.3**	(38)	2s
18–19.5	General	72.5	(31)	77.4	(34)	**74.8**	(32)	1s
	Layer a	69.6	(30)	74.2	(32)	**72.1**	(31)	1s
	Layer b	68.4	(30)	75.5	(33)	**71.8**	(31)	2s
	Buffer 10	69.9	(31)	73.2	(32)	**71.7**	(31)	2s
	Buffer 21	70.5	(31)	74.5	(32)	**72.9**	(31)	3s
19.5–21	General	71.9	(27)	75.4	(28)	**73.2**	(27)	1s
	Layer a	67.5	(25)	72,6	(27)	**70**	(26)	1s
	Layer b	69.9	(26)	75	(28)	**73.2**	(27)	2s
	Buffer 10	69.4	(26)	74.5	(28)	**71.5**	(26)	2s
	Buffer 21	71.6	(27)	74.1	(28)	**72.7**	(27)	2s
21–22.5	General	73.4	(24)	77.6	(25)	**75.7**	(24)	1s
	Layer a	64.1	(21)	72	(23)	**67.1**	(21)	1s
	Layer b	67.9	(22)	75.3	(24)	**72.1**	(23)	1s
	Buffer 10	67	(22)	73.5	(24)	**69.1**	(22)	1s
	Buffer 21	70.2	(23)	74.6	(24)	**72**	(23)	2s
22.5–24	General	69.8	(20)	74	(21)	**71.1**	(20)	1s
	Layer a	66.4	(19)	74.1	(21)	**70.9**	(20)	1s
	Layer b	66.5	(19)	73.1	(21)	**70.7**	(20)	1s
	Buffer 10	63	(18)	71.8	(20)	**69.1**	(19)	2s
	Buffer 21	67.1	(19)	71.8	(20)	**70.5**	(19)	1s
24–25.5	General	69	(17)	75.1	(19)	**72.2**	(18)	1s
	Layer a	67.9	(17)	75.9	(19)	**70.8**	(17)	1s
	Layer b	68.1	(17)	76.6	(19)	**72.2**	(18)	2s
	Buffer 10	69.4	(17)	76.7	(19)	**72.3**	(18)	2s
	Buffer 21	67.7	(17)	76.4	(19)	**71.8**	(17)	1s

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
25.5–27	General	70.6	(16)	73.4	(16)	**72**	(16)	1s
	Layer a	67	(15)	73.1	(16)	**70.5**	(15)	1s
	Layer b	66.9	(15)	72.8	(16)	**70**	(15)	2s
	Buffer 10	62.2	(14)	72.8	(16)	**68.7**	(15)	2s
	Buffer 21	67	(15)	72.7	(16)	**70.2**	(15)	1s
27–28.5	General	65.2	(13)	71.5	(14)	**69.7**	(13)	1s
	Layer a	64.9	(13)	70.8	(14)	**66.9**	(13)	1s
	Layer b	65.2	(13)	71.1	(14)	**68.6**	(13)	2s
	Buffer 10	55.5	(11)	70.7	(14)	**64.1**	(12)	2s
	Buffer 21	64.7	(13)	71.1	(14)	**68.1**	(13)	1s

Medium Intervals

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
9–12	General	77.7	(105)	78.5	(111)	**78**	(107)	3s
	Layer a	72.8	(97)	76.5	(105)	**74.7**	(102)	3s
	Layer b	73	(100)	76.8	(110)	**74.7**	(104)	5s
	Buffer 10	70.8	(97)	76.4	(106)	**74.7**	(102)	7s
	Buffer 21	74.1	(101)	76.8	(106)	**75.5**	(103)	3s
12–15	General	75.9	(61)	78.6	(66)	**77.5**	(64)	3s
	Layer a	73.5	(61)	76.6	(64)	**75.1**	(62)	3s
	Layer b	73.8	(61)	76	(64)	**74.9**	(62)	4s
	Buffer 10	72.1	(60)	76	(64)	**73.7**	(61)	5s
	Buffer 21	73.8	(61)	77.4	(63)	**75.3**	(62)	1s
15–18	General	74.6	(41)	76.8	(44)	**75.5**	(42)	1s
	Layer a	71.5	(39)	75.7	(42)	**74.1**	(41)	2s
	Layer b	70.6	(39)	76.5	(43)	**73.9**	(41)	2s
	Buffer 10	70.3	(39)	76.8	(43)	**73.8**	(41)	3s
	Buffer 21	72.5	(40)	76.7	(42)	**74.7**	(41)	1s
18–21	General	71.7	(28)	76.1	(31)	**73.9**	(29)	1s
	Layer a	69	(27)	75	(30)	**72.7**	(29)	1s
	Layer b	71.4	(29)	74.1	(30)	**72.5**	(29)	2s
	Buffer 10	69.6	(28)	73.9	(30)	**71.8**	(28)	2s
	Buffer 21	70.3	(28)	75.4	(30)	**72.8**	(29)	2s
21–24	General	70.6	(21)	76.5	(24)	**73.6**	(22)	1s
	Layer a	67.4	(21)	72.1	(21)	**69.8**	(21)	1s
	Layer b	68.4	(21)	73.8	(23)	**71**	(21)	1s
	Buffer 10	66.7	(20)	74.3	(22)	**70.9**	(21)	1s
	Buffer 21	68.5	(20)	73.8	(22)	**71.5**	(21)	2s

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
24–27	General	68.6	(16)	74.1	(17)	**71.5**	(16)	1s
	Layer a	67.4	(16)	73.2	(17)	**70.4**	(16)	1s
	Layer b	67.5	(16)	73.1	(18)	**70.5**	(16)	2s
	Buffer 10	63.4	(15)	74.2	(18)	**69.7**	(16)	2s
	Buffer 21	66.6	(16)	73	(17)	**70**	(16)	1s
27–30	General	64	(12)	73.3	(14)	**68.3**	(12)	1s
	Layer a	59.5	(11)	70.8	(13)	**67.7**	(12)	1s
	Layer b	64.8	(12)	70.5	(13)	**68.4**	(12)	1s
	Buffer 10	57.2	(11)	68.9	(13)	**61.4**	(11)	1s
	Buffer 21	57.5	(11)	70	(13)	**66.5**	(12)	1s

Large intervals

Diameter interval	Algorithm	Occupancy						Average time
		Worst case		Best case		Average		
9–19	General	75.6	(52)	78.4	(63)	**77**	(57)	3s
	Layer a	71.8	(49)	77.3	(62)	**74.6**	(56)	3s
	Layer b	72.4	(50)	77.4	(61)	**74.9**	(56)	4s
	Buffer 10	72.9	(50)	76.4	(63)	**74.6**	(56)	5s
	Buffer 21	73.2	(51)	77.4	(64)	**75.8**	(56)	2s
19–29	General	66.8	(17)	75.9	(22)	**71.7**	(18)	1s
	Layer a	67	(17)	75.3	(21)	**71.6**	(18)	1s
	Layer b	66.8	(16)	74.5	(20)	**71.1**	(18)	2s
	Buffer 10	65.3	(17)	73.6	(24)	**70.6**	(18)	2s
	Buffer 21	68.5	(16)	74.8	(21)	**71.2**	(18)	1s
9–29	General	71.1	(25)	77.6	(38)	**74.8**	(29)	1s
	Layer a	63.6	(21)	74.4	(36)	**70.4**	(27)	1s
	Layer b	69.3	(21)	74.6	(34)	**72.6**	(27)	2s
	Buffer 10	61.6	(25)	77	(33)	**71.2**	(29)	2s
	Buffer 21	71.9	(24)	77.5	(35)	**74.6**	(28)	4s