MAREK CHROBAK
MACIEJ ŚLUSAREK

On some packing problem related to dynamic storage allocation

Informatique théorique et applications, tome 22, nº 4 (1988), p. 487-499.
<http://www.numdam.org/item?id=ITA_1988__22_4_487_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 22, n° 4, 1988, p. 487 à 499)

ON SOME PACKING PROBLEM RELATED TO DYNAMIC STORAGE ALLOCATION (*)

by Marek CHROBAK ([1]) and Maciej ŚLUSAREK ([2])

Communicated by J.-E. PIN

Abstract. – *We consider the problem of packing rectangles of height* 1 *when the x-coordinates of their left and right edges are specified. In the off-line case the optimal solution can be trivially found in polynomial time. In the on-line case no algorithm can have performance ratio better than* 3. *A connection between this problem and dynamic storage allocation is shown.*

Résumé. – *On considère le problème de l'empilage de rectangles de hauteur* 1 *lorsque les abscisses des côtés gauches et droits sont spécifiées. On vérifie aisément que dans le cas « off-line » la solution optimale est en temps polynomial. Dans le cas « on-line », il n'existe aucun algorithme dont le coefficient de performance soit meilleur que* 3. *On montre également les rapports de ce problème et du problème de l'allocation dynamique de mémoire.*

1. INTRODUCTION

The two-dimensional bin packing problem has been extensively explored as a model for dynamic storage allocation. Its basic variant assumes that a list of jobs specified by their sizes and residence times is given and the problem is to minimize the make-span; preemptions are not allowed [1, 3, 4]. Geometrically this corresponds to packing rectangles into an infinite height bin so as to minimize the packing height. Vertical coordinate specifies the starting time of a job's execution, horizontal - its address in the memory. A

(*) Received July 1984, revised May 1988.
([1]) Institute of Informatics, Warsaw University, PKiN, p. 850, 00-901 Warsaw, Poland.
([2]) Institute of Informatics, Jagiellonian University, Kopernika 27, 31-501 Cracow, Poland.

special case when the jobs are allocated in the same order as they appear on the list (on-line) is considered in [3, 4].

The second variant allocates jobs strictly on-line, but preemptions are allowed both in time and memory [2]. Yet another on-line version is investigated in [6] where the memory is divided into bins - e. g. disc packages (*see* also a survey [5]).

In this paper we are concerned with a similar model, but for the sake of presentation we adopt an inverse geometrical interpretation. The height (vertical coordinate) corresponds to computer resource (memory) and the horizontal one reflects starting and finishing time of requests. The model looks like follows. There is a list of rectangles of height 1 given on the plane by specifying their left and right edge coordinates and they should be packed by moving them vertically. They cannot intersect and should be considered in the order as they appear on the list – *i. e.* on-line. We call it DPU – dynamic packing of unit blocks, and consider it in terms of a game between an attacker forming a sequence of blocks and a defender whose goal is to pack them. The method is similar to the approach of [10]. The main results can be summarized as follows:

– there is no strategy for DPU with performance ratio better than 3,

– the performance ratio of a natural First Fit strategy is not less than 4,

– assuming that all blocks are of equal length, First Fit has performance ratio 2 and is optimal within a broad class of strategies.

We also show some connection between DPU and the off-line version of the dynamic storage allocation, as defined in [7]. We believe that this can lead to a polynomial time approximation strategy with constant worst-case ratio for the second problem.

The DPU problem can be also considered in graph-theoretic setting, which has been explored by Gyárfás and Lehel [8]. Rectangles may be treated as vertices of an interval graph. Packing them becomes equivalent to graph coloring, and First Fit strategy may be called a greedy method here – for an on-line defined sequence of graph vertices assign to each of them the lowest possible color.

In this setting First Fit performance can be measured as the relation between the number of colors used and the size of the largest clique in the graph. The strongest upper bound [8] (due to W. Just) states that for each interval graph the number of colors used by First Fit is $O(n \log n)$, where n is the size of the largest clique.

2. THE GENERAL VERSION OF DPU

An instance of the *DPU* problem consists of a finite nonempty sequence of blocks, where by a block we mean a pair $v=(f, l)$, $f, l \in Z^+$, $f<l$. The difference $l-f$ is called the length of a block (geometrically, f, l specify the x-coordinates of a unit-height rectangle to be packed). We say that an algorithm A solves *DPU* when given a sequence of blocks $V=(v_i)_{i \leqq n}=((f_i, l_i))_{i \leqq n}$ A packs them by producing a sequence $(a_i)_{i \leqq n}$ of positive integers called addresses, such that

(i) A outputs a_i before reading v_{i+1}, for $i \leqq n-1$,

(ii) if $a_i=a_j$ for some $i<j$ then either $l_i \leqq f_j$ or $l_j \leqq f_i$.

The function a produced by A is called an allocation of V, and $A(V)$ denotes the number of addresses used by A on data V.

Let $OPT(V)$ denote the minimal number of addresses necessary to pack all blocks of V, that is

$$OPT(V)=\min_A A(V).$$

Consider the following algorithm called *FF* (an abbreviation for First Fit).

```
1. $i:=1$;
2. if $i>n$ then STOP;
3. $a_i:=\min\{k \in Z^+ : \forall j<i (a_j=k \Rightarrow [f_i, l_i) \cap [f_j, l_j)=\varnothing\}$;
4. $i:=i+1$; goto 2;
```

Clearly *FF* solves *DPU*; *FF* simply gives v_i the first address unoccupied in the interval $[f_i, l_i)$.

Let us define $D(V)=\max_t |\{i : t \in [f_i, l_i)\}|$, and call it the density of a sequence of blocks V.

THEOREM 1: *For each sequence V of blocks $OPT(V)=D(V)$.*

Proof: Obviously $OPT(V) \geqq D(V)$. Let U denote a permutation of V such that (f_i, l_i) occurs before (f_j, l_j) in U when $f_i<f_j$. Then $OPT(V)=OPT(U)$ and $D(V)=D(U)$. But it is not difficult to observe that $FF(U)=D(U)$. Hence $OPT(V)=D(V)$. ■

Consider a game G_n between an attacker P_1 and a defender P_2 with the following rules:

(1) the i-th move of P_1 is a block $v_i=(f_i, l_i)$,

(2) the response given by P_2 is a number a_i, where $a_1, a_2, \ldots$ is an allocation satisfying (i) and (ii),

(3) the density of the sequence issued by P_1 never exceeds n,

(4) P_1 wins when $P_2(V) \geqq 3n-2$ (where $V=(v_i)_{i=1,2,\ldots}$ and P_2 is also used here to denote the defender's strategy).

LEMMA 1: *For every $n \in Z^+$ there exists a successful strategy S_n for P_1 in G_n.*

Proof: The proof is by induction on n. The strategy S_1 is trivial: P_1 issues the block $v=(1, 2)$. We set $I_1=1$ – the interval length necessary to perform S_1.

Suppose we have a successful strategy S_n for P_1 in G_n and its interval length I_n. Let $t_i=i(I_n+1)+1$ for $i=0$, 1, 2, 3 and $t_4=4I_n+4$. First, P_1 repeats S_n four times in the intervals

$$[t_0, t_1-1), [t_1, t_2-1), [t_2, t_3-1), [t_3, t_4).$$

Denote by V the sequence of blocks issued by P_1 so far and by M_i the set of adresses used by P_2 when playing in the i-th interval, $i=1$, 2, 3, 4. From the inductive assumption we have

(1) $D(V) \leqq n$,

(2) $|M_i| \geqq 3n-2$, for $i=1$, 2, 3, 4,

Let $M=M_1 \cup M_2 \cup M_3 \cup M_4$. For simplicity we describe the remaining part of the game as directed trees; vertices denote P_1's moves and edges are labelled by P_2's responses. *HALT* is the end of the game. There are several cases to be checked. We give the details for those more complicated; the remaining ones are coped with similarly.

(*a*) $|M| \geqq 3n+1$, O.K., P_1 has already won.

(*b*) $|M|=3n$. In this case one P_1's move suffices:

$$(t_0, t_4) \xrightarrow[p \notin M]{} \text{HALT}.$$

(*c*) $|M|=3n-1$.

(c 1) $|M_i|=3n-2$, for $i=1, 2, 3, 4$.

(c 1.1) $M_1 \neq M_2 = M_3 = M_4$. Let $\{x\} = M_1 - M_2$. Then the game can be finished as follows:

$$\begin{tikzcd}(t_3-1, t_4) \arrow[r, "p \notin M"] \arrow[d, "x"'] & (t_0, t_3) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT} \\ (t_2-1, t_3) \arrow[r, "p \notin M"] & (t_0, t_2) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT} \quad (\textit{fig. } 1.a)\end{tikzcd}$$

(c 1.2) $M_1 = M_2 \neq M_3 = M_4$. Let $\{x\} = M_2 - M_3$, $\{y\} = M_3 - M_2$.

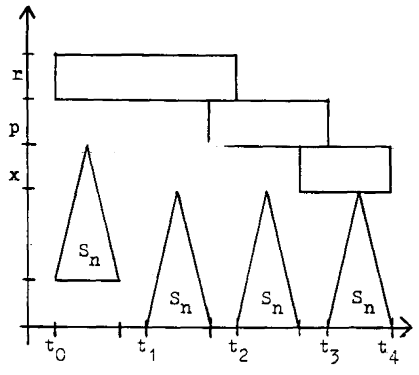

Fig. 1.*a*

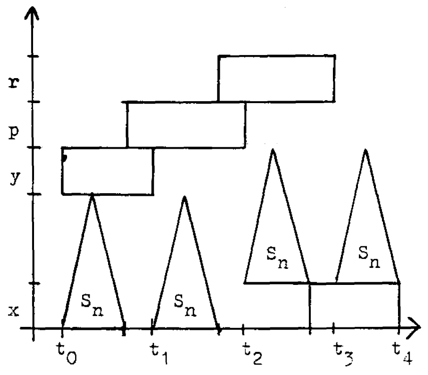

Fig. 1.*b*

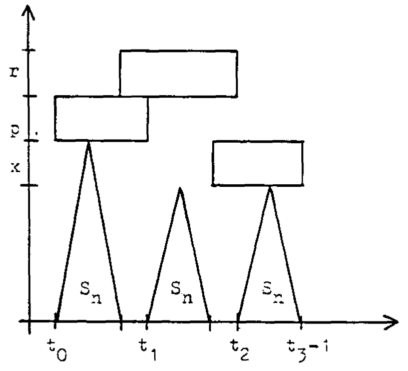

Fig. 1.*c*

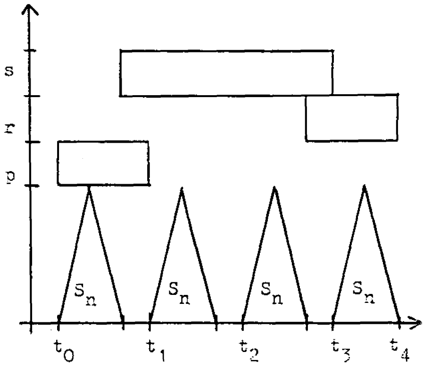

Fig. 1.*d*

The remaining moves are as follows:

$$\begin{tikzcd}
(t_0, t_1) \arrow[r, "p \notin M"] \arrow[d, "y"'] & (t_1-1, t_4) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT} \\
(t_3-1, t_4) \arrow[r, "p \notin M"] \arrow[d, "x"'] & (t_1-1, t_3) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT} \\
(t_1-1, t_2) \arrow[r, "p \notin M"] & (t_2-1, t_3) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT}
\end{tikzcd}$$
(*fig.* 1.*b*).

The remaining subcases of (c 1) are:

$$M_1=M_2=M_3 \neq M_4, \qquad M_1=M_3=M_4 \neq M_2,$$
$$M_1=M_2=M_4 \neq M_3, \qquad M_1=M_4 \neq M_2=M_3, \qquad M_1=M_3 \neq M_2=M_4.$$

(c2) $|M_j|=3n-1$ for exactly one $j \in \{1, 2, 3, 4\}$. Let $j=1$ (for $j=2$, 3, 4 the argument is similar).

(c2.1) $M_2=M_3$. Let $\{x\}=M_1-M_2$. The following moves lead to the victory of P_1:

$$\begin{tikzcd}
(t_2-1, t_3) \arrow[r, "p \notin M"] \arrow[d, "x"'] & (t_0, t_2) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT} \\
(t_0, t_1) \arrow[r, "p \notin M"] & (t_1-1, t_2) \arrow[r, "r \notin M \cup \{p\}"] & \text{HALT}
\end{tikzcd}$$
(*fig.* 1.*c*)

(c2.2) $M_2 \neq M_3$. This case is simple:

$$(t_0, t_1) \xrightarrow{p \notin M} (t_1-1, t_3) \xrightarrow{r \notin M \cup \{p\}} \text{HALT}$$

(c3) There are at least two $j_1, j_2 \in \{1, 2, 3, 4\}$ such that $|M_{j_1}|=|M_{j_2}|=3n-1$. The moves of P_1 are similar to those from (c2.2).

(d) $|M|=3n-2$. All M_i's are equal and the moves of P_1 are as follows:

$$\begin{tikzcd}
(t_0, t_1) \arrow[r, "p \notin M"] & (t_3-1, t_4) \arrow[r, "r \notin M \cup \{p\}"] \arrow[d, "p"'] & (t_1-1, t_3) \arrow[r, "s \notin M \cup \{p, r\}"] & \text{HALT} \\
& (t_1-1, t_2) \arrow[r, "r \notin M \cup \{p\}"] & (t_2-1, t_3) \arrow[d, "s \notin M \cup \{p, r\}"] & \\
& & \text{HALT} &
\end{tikzcd}$$
(*fig.* 1.*d*)

In each case P_2 is forced to use at least $2n+1$ addresses. Since the density of the instance increases at most by one, the above procedure gives us the strategy S_{n+1}. We finish the proof by setting $I_{n+1}=t_4-t_0=4I_n+3$. ■

As a consequence we obtain:

THEOREM 2: *There does not exist a strategy A for DPU such that for some $\varepsilon > 0$ and δ*

$$A(V) \leqq (3-\varepsilon) D(V) + \delta$$

for every V in DPU.

Proof: We have $(3n-2)/n > 3 - \varepsilon + \delta/n$ for sufficiently large n, so the thesis follows from lemma 1. ■

LEMMA 2: *For each $n>0$ there exists $V_n \in DPU$ such that $D(V_n) = n$ and $FF(V_n) \geqq 4n-9$.*

Rroof: We construct V_n's inductively. First we show that there are $V_1, \ldots, V_7$ such that $FF(V_n) \geqq 3n-2$ for $n = 1, \ldots, 7$. Any single block can serve as V_1. For $n = 2, \ldots, 7$ V_n is formed as a concatenation of four separate V_{n-1}'s with four new blocks added, as depicted in figure 2*a*. This yields $FF(V_n) = FF(V_{n-1}) + 3$ for $n = 2, \ldots, 7$. Therefore

$$FF(V_n) \geqq 3n-2 \geqq 4n-9 \qquad \text{for} \quad n = 1, \ldots, 7.$$

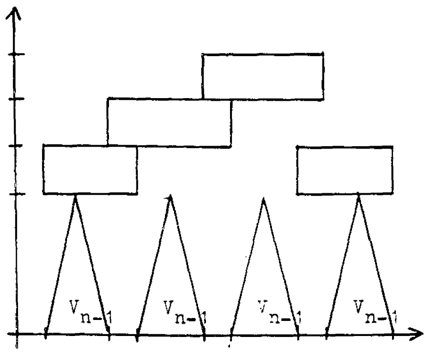

Fig. 2.*a*

For $n \geqq 8$ the construction of V_n is more complicated. Using figure 2*b* we describe V_n as follows. First, eight V_{n-4} instances, four V_{n-3}, four V_{n-2} and two V_{n-1} are concatenated, each one defined in its own sufficiently long interval. Then comes a sequence of 36 blocks whose location is depicted explicitly in figure 2*b*. First four blocks are those which obtained address $FF(V_{n-4}) + 1$, the next two in the sequence are the ones placed in the address

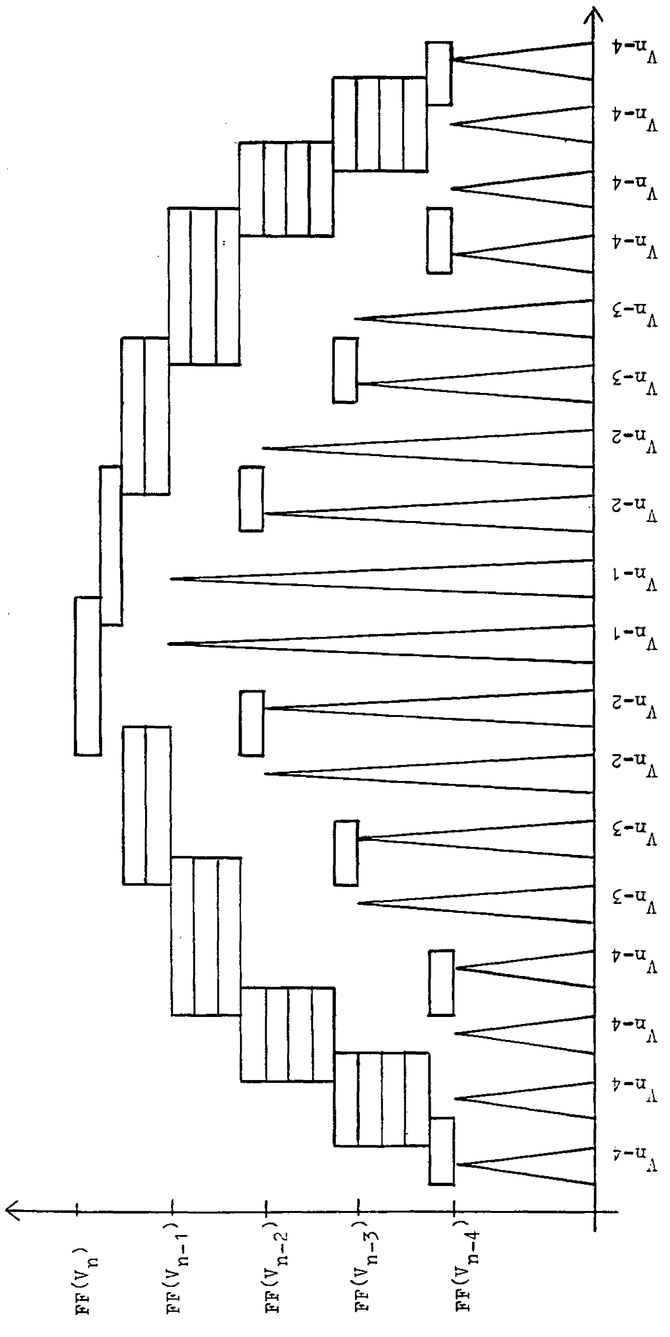
V_{n-4}
V_{n-4}
V_{n-4}
V_{n-4}
V_{n-3}
V_{n-3}
V_{n-2}
V_{n-2}
V_{n-1}
V_{n-1}
V_{n-2}
V_{n-2}
V_{n-3}
V_{n-3}
V_{n-4}
V_{n-4}
V_{n-4}
V_{n-4}
$FF(V_n)$
$FF(V_{n-1})$
$FF(V_{n-2})$
$FF(V_{n-3})$
$FF(V_{n-4})$

Fig. 2.*b*

$FF(V_{n-4})+2$ and so on. The last block in the sequence (and in the whole instance V_n) is the one which has been given the highest address $FF(V_n)$.

Assuming $n \geqq 8$ one can easily see that the density of V_n is exactly n, and $FF(V_n)=FF(V_{n-1})+4$. This finishes the proof of the induction step, from which the thesis follows. ■

THEOREM 3: *There do not exist* $\varepsilon>0$ *and* δ *such that*

$$FF(V)<(4-\varepsilon)D(V)+\delta$$

for every V *in* DPU.

Proof: Immediate from Lemma 2. ■

3. A SUBPROBLEM OF DPU

Suppose we require that all blocks are of the same length. This restricted version is called *UDPU* (*U* from "uniform").

THEOREM 4: *For every* $V=(v_i)_{i\leqq n}$ *in* $UDPU\ FF(V)\leqq 2D(V)-1$. *Moreover, the ratio* 2 *is tight.*

Proof: Let $v_i=(f_i, l_i)$ be such a block that $a_i=FF(V)$. We define

$$V_1=\{v_j : f_i<l_j<l_i\},\qquad V_2=\{v_j : f_i<f_j<l_i\},$$
$$V_3=\{v_j : f_j=f_i\}.$$

Then

$$\begin{aligned}FF(V)=a_i&\leqq |V_1|+|V_2|+|V_3|+1\\ &\leqq |V_1|+|V_3|+1+|V_2|+|V_3|+1-1\leqq 2D(V)-1.\end{aligned}$$

This gives the upper bound. To show that this ratio is tight we construct the following instances of *UDPU*:

$$V_n=(v_{n,1}, \ldots, v_{n,n}, u_{n,1}, \ldots, u_{n,n-1}, v_{n,n+1}, \ldots, v_{n,2n-1})$$

for $n\in Z^+$, where

$$v_{n,i}=(i, n+i)\quad\text{for } i=1, \ldots, 2n-1,$$

and $u_{n,i}=(2n+i-1, 3n+i-1)$ for $i=1, \ldots, n-1$ (*fig.* 3). Then $D(V_n)=n$ and $FF(V_n)=2n-1$, hence $FF(V_n)/D(V_n)\to 2$ as $n\to\infty$. ■

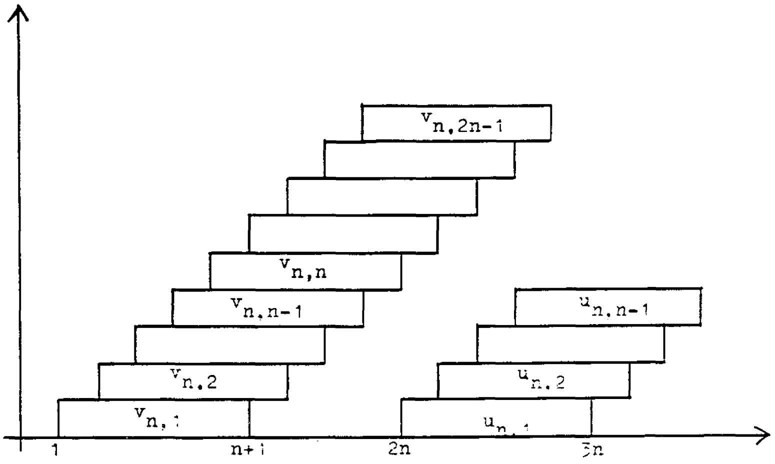

Fig. 3

Proving a nontrivial lower bound for all packing strategies in case of UDPU seems to be much harder than for the general version. We can only show that FF is optimal within quite a wide range of algorithms.

THEOREM 5: *Assume that an algorithm A solving UDPU obeys the rule:*
(★) *never choose a new address if there is room in one already used.*
Then the performance ratio of A is not less than 2.

Proof: Consider V_n of the proof of theorem 4. Let

$$U_n=(v_{n,1}, u_{n,1}, \ldots, v_{n,n-1}, u_{n,n-1}, v_{n,n}, \ldots, v_{n,2n-1})$$

for $n \in Z^+$. Then, if A satisfies (★) $A(U_n)=FF(U_n)=2n-1$. ■

4. DPU AND DYNAMIC STORAGE ALLOCATION

Consider the following dynamic storage allocation problem—DSA in short. There is a collection

$$T=\{b_i=(r_i, d_i, s_i), i=1, \ldots, n\}$$

of memory requests (blocks) given, where $r_i, d_i, s_i \in Z^+$ denote respectively the arrival time, departure time and the size of the i-th block, $r_i<d_i$,

$i=1, \ldots, n$. We ask for the smallest memory size M such that there exists an allocation function

$$a: \{1, \ldots, n\} \rightarrow Z^{+} \cup \{0\}$$

satisfying:

(1) $a_i+s_i \leqq M$, for $i=1, \ldots, n$,

(2) for each $i \neq j$ $(a_j \leqq a_i < a_j+s_j \Rightarrow r_i \geqq d_j \vee r_j \geqq d_i)$.

The answer should be constructive, *i. e.* explicitly giving the numbers a_i. The problem is NP-hard in the strong sense even when $s_i \in \{1, 2\}$ [7]. Restricting residence times d_i-r_i to be not greater than 3 (they may be uniform) and allowing sizes to be unlimited also preserves NP-hardness [12].

We do not know any approximation strategy for DSA having constant performance ratio. Using Robson's results [10] it can be proved that no on-line algorithm (here "on-line" means that the blocks should be packed in order according with their arrival times), can achieve such a ratio [11], as well as quite a few off-line ones. We believe that the strategy described below possesses the required property.

Assume that all block sizes are powers of two. The Buddy-Decreasing-Size algorithm (BDS in short) works in two stages:

(1) sort the blocks on descending size,

(2) place each block as low as possible, considering the blocks in the order from (1).

Denote by $BDS(T)$ the smallest memory size needed to perform BDS on T and by $OPT(T)$ the minimum over all packing strategies. $D(T)$ is defined as

$$D(T)=\max_{t} \sum_{i: r_i \leqq t < d_i} s_i$$

and is called the density of T. Unlike in DPU we may have that $OPT(T)>D(T)$ for some instances T of DSA. In general only the inequality holds: $OPT(T) \geqq D(T)$.

THEOREM 6: *For every V in DPU there exists T in DSA such that $D(T)=OPT(T)$ and*

$$\frac{FF(V)}{D(V)}=\frac{BDS(T)}{D(T)}.$$

And conversely, for each T in DSA there exists V in DPU such that $D(V)=D(T)$ and $FF(V)=BDS(T)$.

Proof: (1) Let $V=(v_i)_{i \leqq n}$ and $h=FF(V)$. Construct T in the ffollowing way: Replace each block v_i by k_i identical blocks $b_{j_l}=(r_{j_l}, d_{j_l}, s_{j_l})$ where $k_i=2^{a_i-1}$, $r_{j_l}=f_i$, $d_{j_l}=l_i$, $s_{j_l}=2^{h-1}/k_i$ and a_i is the allocation generated by FF (here, as usual f_i, l_i are such that $v_i=(f_i, l_i)$). BDS, given T as the input, produces a configuration of the same shape, and $OPT(T)=D(T)$ follows from Theorem 1.

(2) Let T be a sequence of DSA blocks, their sizes being powers of 2, sorted on descending size. Preserving this order, replace each block $b_i=(r_i, d_i, s_i)$ by s_i identical DPU blocks $v_{i_j}=(r_i, d_i)$. Observe that when BDS locates a block b_i there is no gap in the interval $[r_i, d_i)$ below a_i (if it were, it would be of size not less than s_i because of descending order of sizes). Hence v_{i_j} is assigned the same address by FF as it would get if treated as a part of b_i. ■

COROLLARY 1: *For any $r<4$ there are instances T in DSA of arbitrarily large density such that $BDS(T)/OPT(T)>r$.* ■

Using the idea of the buddy-system [9] the domain of BDS strategy can be easily extended to include blocks of any sizes, at the cost of doubling the ratio.

Observe that Theorem 6 is valid when we impose restrictions on block lengths. Let $UDSA$ consist of those instances of DSA in which all blocks are of equal length. So we obtain:

COROLLARY 2: *The strategy BDS for $UDSA$ has the asymptotic performance ratio equal to* 4. ■

Note that $UDSA$ is NP-complete, even when all blocks are required to be of length 3 [12]. Therefore Corollary 2 gives a polynomial-time approximation strategy for a simple, but still NP-complete, version of DSA.

5. FINAL REMARKS

The most intriguing question left open is that of tight bound for the performance of FF strategy for DPU. A complicated extension of the proof of Lemma 2 yields that its ratio is not less than 4.4. Note that similarly as in Corollary 2 showing that this ratio is constant (which we believe is the case—though the ratio 4 from Theorem 3 is much greater than we expected at first) we would prove that BDS is an approximation algorithm for DSA with constant ratio. Clearly this ratio would be too large to allow any practical applications. However, once it is known that such an approximation is possible at all, it may be easier to find more efficient strategies for DSA. Moreover, the notion of "approximability" is useful in classifying hard (in particular, NP-complete) problems according to their difficulty. The easiest problems are those approximable with the ratio $1+\varepsilon$, where ε is arbitrarily small. Then come the problems approximable with constant ratios much greater than 1. At last, the most difficult problems are those which cannot be approximated with any constant ratio unless $P=NP$—for example the travelling salesman problem [7].

REFERENCES

1. B. S. Baker, D. J. Brown and H. P. Katseff, *A 5/4 Algorithm for Two-Dimensional Packing*, J. Algorithms, Vol. 2, 1981, pp. 348-368.
2. B. S. Baker and E. G. Coffman Jr., *A Two-Dimensional Bin-Packing Model of Preemptive FIFO Storage Allocation*, J. Algorithms, Vol. 3, 1982, pp. 303-316.
3. B. S. Baker and J. S. Schwartz, *Shelf Algorithms for Two-Dimensional Packing Problems*, SIAM J. Comput., Vol. 12, 1983, pp. 505-525.
4. D. J. Brown, B. S. Baker and H. P. Katseff, *Lower Bounds for On-Line Two-Dimensional Packing Algorithms*, Acta Informatica, Vol. 18, 1982, pp. 207-225.
5. E. G. Coffman Jr., *An Introduction to Combinatorial Models of Dynamic Storage Allocation*, SIAM Review, Vol. 23, 1983, pp. 311-325.
6. E. G. Coffman Jr., M. R. Garey and D. S. Johnson, *Dynamic Bin Packing*, SIAM J. Comput., Vol. 12, 1983, pp. 227-258.
7. M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, San Francisco, 1979.
8. A. Gyarfás and J. Lehel, *On-Line and First-Fit Colorings of Graphs, Computer and Automation Institute of the Hungarian Academy of Sciences*, preprint, 1986.
9. D. E. Knuth, *The Art of Computer Programming*, Vol. 1, Fundamental Algorithms, 2nd ed., Addison-Wesley, Reading 1973.
10. J. M. Robson, Bounds for some functions concerning dynamic storage allocation, *JACM*, Vol. 21, 1974, pp. 491-499.
11. M. Slusarek, *An off-line storage allocation algorithm*, Info. Proc. Lett., Vol. 24, 1987, pp. 71-75.
12. M. Slusarek, *NP-Completeness of Storage Allocation*, Jagiellonian University Scientific Papers, Computer Science Series, Vol. 3, 1987, pp. 8-18.