**Seminar of I. Vekua Institute**
**of Applied Mathematics**
**REPORTS, Vol. 35, 2009**

# ONLINE TOOL TO FIND THE BOUNDS OF OBJECTIVE FUNCTIONS FOR A CLASS OF ONE-DIMENSIONAL BIN PACKING PROBLEMS

Fedulov G., Rukhaia K., Tibua L., Gulua K., Iashvili N.

**Abstract**. We research a class of 16 combinatorial models, that are semantically near to a known One-Dimensional Bin Packing task. All models have a large number of practical applications in the different areas. A general description of class is to divide an initial set of weights into a some number of disjoint subsets with the given properties. Primary attention of paper has been given to the estimation of quality of approximation solutions as a measure of closeness to the optimal solutions. With that purpose, we build the fast bounds of objective function which the approximation solutions are compared with. To find the bounds, we use two main blocks: an initial reduction and estimation corridor. Our algorithms can be used in practice for large-sizes tasks as an alternative to other approaches when the time factor is important. We offer our estimation approach as the project decisions to develop an online mobile program tool in C#2008, ASP.NET 3.5 and SQL Server 2005 for the mass users to use in the Internet without any special mathematical knowledge.

**Keywords and phrases**: Bin packing, approximation solution, lower bound, upper bound, initial reduction, estimation corridor.

**AMS subject classification (2000):** 90C10.

We research a class of combinatorial models [1-6] that are semantically near to the known One-Dimensional Bin Packing Problem (1DBPP). All models have a practical applications in the different areas: One-Dimensional Stock Cutting, placing of files on CDs, Scheduler Theory, a Container Loading and so on. A general description of class is following. Given a set of items $A = \{a_1, a_2, \ldots, a_n\}$, to each item $a_k$ corresponds a weight $s(a_k)$ and a profit(cost) $p(a_k)$, $s(a_k) \geq s(a_{k+1})$. We need to divide the initial set $A$ into $M$ disjoint subsets $A_1, A_2, \ldots, A_M$, $\bigcup_{i=1}^{M} A_i = A$, $A_i \cap A_j = \emptyset$, $i \neq j$, $i, j \in [1, M]$ with the given properties. All subsets are independence ones and a sequence of weights within each subset is any. We denote $S(A) = \sum_{k=1}^{n} s(a_k)$ as a sum of weights $A$, $C_i = \sum_{a_k \in A_i} s(a_k)$ as a sum size of items (a bin content) of $i$th bin and $P_i = \sum_{a_k \in A_i} p(a_k)$ as a sum profit(cost) of items of $i$th bin, $i \in [1, M]$. One can represent an initial set of weights $\{s(a_1), s(a_2), s(a_n)\}$ in a compact form: $W = \{w_1 \circ k_1, w_2 \circ k_2, \cdots, w_m \circ k_m\}$, where $w_1 > w_2 > \cdots > w_m$ , $w_i \circ k_i$ is a group of equal weights $w_i$ , $k_i$ is a multiplacity, $\sum_{i=1}^{m} k_i = n$, $\sum_{i=1}^{m} k_i w_i = S(A)$. Thus, a parameter $m$ is a **number of different weights**. Below we give a description of models of **1DBP** class.

**Model 0. Base Model.** Given a fixed list of bins $L = \{B_1, B_2, \ldots, B_M\}$, $B_i \geq B_{i+1}$, the $B_i$ is a capacity of $i$th bin, $S(L) \geq S(A)$, where $S(L) = \sum_{i=1}^{M} B_i$, $S(A) = \sum_{k=1}^{n} s(a_k)$. We need to pack $A$ into $L$: $C_i \leq B_i$, $C_i = \sum_{a_k \in A_i} s(a_k)$ is a sum size of items (a bin content) of $i$th bin, $i \in [1, M]$. An answer is **YES** if we can pack $A$ into $L$ and **NO** otherwise.

**Model 1. Classical Bin Packing.** To divide $A$ into a **minimal number** $M$ of disjoint subsets: $C_i \leq B$, $i \in [1, M]$, $B$ is a bin capacity.

**Model 2. Bin Covering.** To divide $A$ into a **maximal number** $M$ of disjoint subsets: $C_i \geq B$, $i \in [1, M]$, $B$ is a bin quota.

**Model 3. Bin Packing & Bin Covering 1.** To divide $A$ into **a minimal number** $M$ of disjoint subsets: $B_{\min} \leq C_i \leq B_{\max}$, $i \in [1, M]$, where the parameters $B_{\min}$ and $B_{\max}$ are the lower and upper thresholds respectively.

**Model 4. Bin Packing & Bin Covering 2. Model 4** is similar to **Model 3** but it is need to find **a maximal number** $M$.

**Model 5. Schedule Theory.** $M$ is fixed. To find a **minimal bin size** $B$ in order to divide $A$ into $M$ of disjoint subsets: $C_i \leq B$, $i \in [1, M]$.

**Model 6. Schedule Theory (General Model 5).** Given a list $\tau_1, \tau_2, \dots \tau_M$ of positive real numbers. It is need to find a **minimal positive integral number** $T$ in order to pack $A$ into a list of bins $L = \{B_1, B_2, \dots B_M\}$: $C_i \leq B_i$, $i \in [1, M]$, $B_i = T\tau_i$.

**Model 7. Bin Packing with a range of B.** Given a range $[B_{\min}, B_{\max}]$ of bin capacities. It is need to find an **optimal bin capacity** $B$ in order to a product $MB \to min$ , where $M$ is a solution of **Model 1**.

**Model 8. Bin Packing with the decreasing bin capacities.** Given a decreasing sequence of bins $B_1 \geq B_2 \geq \dots \geq B_q$. It is need to find a minimal number $M \leq q$ in order to pack $A$ into a list of bins $\{B_1, B_2 \dots B_M\}$: $C_i \leq B_i$, $i \in [1, M]$.

**Model 9. Maximal loading of weights.** Given a fixed list of bins $L = \{B_1, B_2 \dots B_M\}$, where $S(A) \geq S(B)$, where $S(B) = \sum_{i=1}^{n} B_i$. It is need to find a subset $A' \subseteq A$ in order to pack $A'$ into $L$: $C'_i \leq B_i$, $i \in [1, M]$ and a sum weight $S(A') \to max$.

**Model 10. Maximal loading of profits (General model 9). Model 10** is similar to **Model 9** but it is need to find a subset $A'$: sum profit $S(P') \to max$.

**Model 11. Minimal loading of weights Model 11** is similar to **Model 9** but it is need to find a subset $A'$: $C'_i \geq B_i$, $i \in [1, M]$ and a sum weight $S(A') \to min$.

**Model 12. Minimal loading of costs (General Model 11). Model 12** is similar to **Model 11** but it is need to find a subset $A'$: a sum cost $S(P') \to min$.

**Model 13. Minimal sum capacity of subset of bins.** Given a list of bins $L = \{B_1, B_2, \dots B_M\}$, where $S(A) \leq S(L)$. It is need to find a subset $L' \subseteq L$ in order to pack $A$ into $L'$: a sum bin capacity $S(L') \to min$.

**Model 14. Minimal sum cost of subset of bins (General Model 13)**. Given a list of bins $L = \{B_1, B_2, \dots B_M\}$. Each bin $B_i$ has a cost $P_i$. **Model 14** is similar to **Model 13** but it is need to find a subset $L'$: a sum bin cost $S(\ P') \to min$.

**Model 15. Bin Packing with a range of multiplicities of weight.** We consider such $W$, where $k_i \in [k_i^{min}, k_i^{max}]$. We fix $k_i$ and for a given bin capacity $B$ and solve **Model 1**. We need to find such $k_i$ in order a sum waste $MB - S(W) \to min$, where $S(W) = \sum_{i=1}^{m} w_i k_i$.

All models one can lead to **Model 0** as **NP-complete** in process of solving. These models are the **NP-hard** problems to find the optimal solutions for the arbitrary initial data and are solved in practice as rule using the approximation algorithms that it is necessary to evaluate somehow. In this case we find the bounds of objective function:

a lower bound $LB(A)$ for the tasks "to minimum" and an upper bound $UB(A)$ for the tasks "to maximum". One can write "$UB(A) =$ *approximation solution*" for the tasks "to minimum" and "$LB(A) =$ *approximation solution*" for the tasks "to maximum". Thus, we get $LB(A) \leq OPT(A) \leq UB(A)$ for the both cases. Since $OPT(A)$ is not known, we consider a value $p = ((UB(A) - LB(A))/LB(A)) \cdot 100\%$ as a measure of closeness to $OPT(A)$. In case $p = 0$ we claim "*approximation solution = optimal solution*". A finding of both fast and quality bounds has a practical importance especially for the tasks of large parameters $m$ that is a large problem to get the fast bounds for the modern algorithms (e.g. for a known-well Linear Programming approach [3]). In practice often arrives a problem to find the fast bounds of objective function during for a given time limit. Because of is an actual problem to make the sets of different bounds $LB_i(A)$ and $UB_i(A)$ to have a choice. We offer an estimation technology to form the fast bounds of objective functions for our models. This technology can be used as base to make the bounds of objective functions for the other models that use an idea to divide the initial set $A$ into the disjoint subsets with the given properties. The technology is of the two blocks: the initial reduction and estimate corridor. These blocks are interlinked closely. The results of the first block are used in the second block and vice versa.

The first block removes the dominate groups of weights from the initial data and produces the initial reduction of two types. The first type (**A-type**) is used only for **Model 1** by a formula: $OPT(A) = M_0 + OPT(A'), M_0 = M_1 + M_2 + M_3 + M_4 + \cdots + M_H$, where $M_1$, $M_2$, $M_3$, $M_4$, $\ldots M_H$ are the numbers of the dominate singletons, pairs, triplets, quarters, $\ldots$ respectively, $M_0$ is a number of bins reduced, $A' = A \setminus A^0, A^0 = \bigcup_{i=1}^{H} A^i$, $|A^0| = M_1 + 2M_2 + 3M_3 + 4M_4 + \cdots + HM_H$, $A^i = \bigcup_{j=1}^{M_i} A^i_j, H := H(B)$ is a maximal number of weights to put into a bin of capacity $B$. A singleton is a bin of one weight, a pair is a bin of the two weights, a triplet is a bin of the three weights and so on. Each subset $A^i_j$ is a dominate group of $i$ weights. Here $A^1, A^2, A^3, A^4, \ldots A^H$ are the lists of the dominate singletons, pairs, triplets, quarters, $\ldots$ respectively. We call a group $G = \{a_{N_k(i)}\}$, $N_k(i) = N_{k-1}(i) + 1$, $k \in [1, i]$ as a dominate one, if a number $p := N_1(i)$ has a property: $\sum_{k=p}^{p-k+1} s(a_k) \leq B$, $\sum_{k=p-1}^{p+i-2} s(a_k) > B$, where $N_0(i) := N_1(i) - 1$ is a number of items before $a_p$. Here $N_1(1)$ defines a number for the dominate singletons, $N_1(2)$ for the dominate pairs, $N_1(3)$ for the dominate triplets, $N_1(4)$ for the dominate quarters and so on. If an optimum solution has at least one group $G' = \{a_{N'_k(i)}\}, k \in [1, i]$, where $N'_1(i) \geq N_1(i)$, then we can remove $G$ from $A$ and put $G$ into $A^0$ since $s(a_{N_k(i)}) \geq s(a_{N'_k(i)})$ because of $N'_k(i) > N_k(i)$, $k \in [1, i]$. A main idea to find $G' = \{a_{N'_k(i)}\}$ is following. We want to prove a fact: there are exist an optimal solution that has at least one group of $i$ items $G' = \{a_{N'_k(i)}\}$, $N'_{k+1}(i) > N'_k(i)$, $k \in [1, i]$, with a property $N'_1(i) > N_1(i)$ where $s(a_{N_k(i)}) \geq s(a_{N'_k(i)})$ , $k \in [1, i]$ . If we prove this fact then a group $G = \{a_{N_k(i)}\}$ dominates a group $G' = \{a_{N'_k(i)}\}$ therefore we can remove a dominate group $G = \{a_{N_k(i)}\}$ from the initial set $A$. To recognize the dominate groups of weights we developed the fast reduction algorithms A1, A2 and A3 for A-type [5,6]. Below we give a brief description these algorithms.

**Algorithm A** to build $A^0$.
**1.** $A^0 := \emptyset$, $A' := A$, $M :=$**P**$(A')$.

**2.** $i := 0$, $\mu(0) := 0$.
**3.** $i := i + 1$.
**4.** $M' := M - \mu(i-1)$. **If** ( $M' = 0$ ) **STOP**.
**5.** Algorithms **A2** and **A3** to find $G$.
**6. If** $G \neq \emptyset$ { $A^0 := A^0 \bigcup G$, $A' := A' \setminus G$, $M :=$**P**$(A')$ }.
**7.** Algorithm **A1** to find $\mu(i)$.
**8. Go to 2**.

Here **P**$(A')$ is an algorithm that produces a bound $M : \min_M P(A', B^{\min}, B^{\max})$=YES, $B_i^{\min} = w_m, B_i^{\max} = B, i \in [1, M]$, $\mu(i-1)$ is a maximal number of bins that can be used by weights of range $[1, N_0(i)]$, $\mu(i-1) \leq N_0(i)$. The algorithms **A1** and **A2** try to find $G$.

**Algorithm A1** to find $\mu(i)$. We will find $\mu(i)$ by using a formula $\mu(1) = N_0(2)$, $\mu(i) = \mu(i-1) + x(i)$, $i \geq 2$, where $x(i)$ is a maximal number of bins that can use $x(i)$ weights from a range $\Delta(i) = [N_1(i), N_0(i+1)]$, $x(i) \leq k_0 = N_0(i+1) - N_1(i) + 1$. We ask: can we put each weight of $\Delta(i)$ into a personal bin? Suppose we have put $k$ weights of $\Delta(i)$ into $k$ bins. We consider a sum of the first $k$ weights of $\Delta(i)$ as $S_1(k) = \sum_{j=N_1(i)}^{N_1(i)+k-1} s(a_j)$ and a sum of $ik$ easiest weights as $S_2(k) = \sum_{j=n-ik+1}^{n} s(a_j)$. If $S_1(k) + S_2(k) > kB$ then at least one of $k$ bins will be have not more $i$ weights. As any group $\{s(a_{J_1}), s(a_{J_2}), \dots s(a_{J_i})\}$ is dominated by $G$, $J_1, J_2, \dots, J_i \in \Delta(i)$, we can not use $k$ bins by $k$ weights of $\Delta(i)$. Because of we have the two cases: to put the $k$th weight of $\Delta(i)$ into a bin of $\mu(i-1)$ bins where we have put the weights $s(a_j)$, $j \in [1, N_0(i)]$ or to join the $k$th weight with one of previous $k-1$ weights $s(a_j)$, $j \in [\mu(i-1)+1, \mu(i-1)+k-1]$. The other details we put to an algorithm of building $x(i)$.

We denote $S_1(i, q) = \sum_{j=n-iq+1}^{n} s(a_j)$ and $S_2(q) = qB$.

**Algorithm to find** $x(i)$
**1.** $x(i) := 0$, $k_0 := N_0(i)$, $k := k_0$, $p := 0$.
**2.** $k := k + 1$, $q = k - k_0$. **If** ( $k > N_0(i+1)$ ) **STOP**
**3.** If $\left( \sum_{j=k_0+1}^{k} s(a_j) + S_1(i, q) \leq S_2(q) \right)$ $\{x(i) := x(i) + 1\}$
**else** { $p := p + 1$, $k_0 := k_0 + 1$, **If** ( $p = i$ ) $\{x(i) := x(i) + 1, p := 0\}$ }.
**4. Go to 2**.

**Algorithm A2** to build $A^0$. We consider a number $M'$ of bins of range $[1, \mu(i-1)]$. Let an algorithm packs a maximum number $K$ of the weights $s(a_j)$ into $M'$ bins, $K \geq M'$, $j \in [1, K]$. It follows we can put not more $n-K$ weights into $M-M'$ bins since a set of weights $\{s(a_1), \dots, s(a_K)\}$ dominates any set of $K$ weights $\{s(a_{J_1}), \dots, s(a_{J_k})\}$ since $s(a_k) \geq s(a_{J_k})$, $k \in [1, K]$. If $n - K < (i+1)(M - M')$ it follows we find at least one group of $i$ weights to put into a bin from $M - M'$ bins. If we get a result $n - K < (i+1)(M - M')$ for all $M' \in [1, \mu(i-1)]$ then we can remove the dominate group $G$ from $A$ and put $G$ into $A^0$.

**Algorithm A3** to build $A^0$. Let a difference $M'' = M - \mu(i-1) > 0$. It follows: each bin of range $\Delta = [N_1(i), N_1(i) + M'' - 1]$ has the weights with the numbers $j \geq N_1(i)$. We consider any $k \in \Delta$ and ask: can we put $k$ weights $\{s(a_j)\}, j \in [N_1(i), N_1(i)+k-1]$

into $k$ bins? In other words: can we put only one weight into each bin? In this case each bin have to get not less $i+1$ weights. We denote $S_1(k) = \sum_{j=N_1(i)}^{N_1(i)+k-1} s(a_j)$ as sum of $k$ weights and $S_2(k) = \sum_{j=n-ik+1}^{n} s(a_j)$ as sum of $ik$ easiest weights. If $S_1(k)+S_2(k) > kB$ then at least one of $k$ bins must get a group of $i$ weights of range $\Delta$. As any $i$-group $\{s(a_{j_1}), s(a_{j_2}), \dots s(a_{j_i})\}$ is dominated by $\{s(a_{N_1(i)}, s(a_{N_1(i)+1}, \dots s(a_{N_1(i)+i-1})\}$, $j_1, j_2, \dots j_i \in \Delta$, we claim: we can't put $k$ weights into $k$ bins. Now we want to know: can we put $k$ weights $s(a_j)$ into $k' \in [1,k]$ bins? Again, each bin have to get not less $i+1$ weights. We denote $S_2(k') = \sum_{j=n-k'(i+1)+k+1}^{n} s(a_j)$. If $S_1(k) + S_2(k') > k'B$ then at least one of $k'$ bins gets not more $i$ weights of range $\Delta$. If we get a result $S_1(k) + S_2(k') > k'B$ for all $k' \in [1,k]$ for a fixed $k \in [1, M'']$, then we can remove the dominate group $G$ from $A$ and put $G$ into $A^0$.

The second type (**B-type**) is the general one for all models and is used to solve **Model 0** by a formula: $(A, L) \to (A', L')$. Here we lead an initial data $(A, L)$ to a data $(A', L')$. Here we try to find a dominate group $G = \{a_{N_k(i)}\}$ for a range of bins $[B_{K_1}; B_{K_2}]$, $1 \le K_1 < K_2 \le M$: $\sum_{k=1}^{i} s(a_{N_k}) \le B_{K_2}$, $N_1' \ge N_1$, where $N_1$ we find from $s(a_{N_1}) \le B_{K_1}$, $s(a_{N_1-1}) > B_{K_1}$ . We developed the fast algorithms B2 and B3 for B-type [6] too. Here our algorithms solve more difficult problem to recognize by algorithms A1, A2, A3. Below we give a brief description these algorithms B2 and B3, Let we given by the constraints $B_i \ge B_i^{min}, i \in [1, M]$. We will use a parameter **par** as **1** in case $B^{min} \ne \emptyset$ and as **0** otherwise. Now we consider a group $G$ and a range of bins $B_i$, $i \in [q, Q]$. Let $P(q)$ is a minimal number: $s(a_{P(q)}) \le B_q$, $s(a_{P(q)-1}) > B_q$. Let $\sum_{j=p}^{p+i-1} s(a_j) \le B_Q$, $p := N_1(i)$, here $N_1(i)$ we form for $B := B_Q$, $i = 1, 2, \dots H(B)$. Let a difference $M'' = Q - q + 1 - \mu(i-1) > 0$.

**Algorithm B2.** We consider a number $M'$ of bins of range $[1, \mu(i-1)]$. We build $B'$ as a set of bins as following: $B_i' = B_i$, $i \in [1, q-1]$, $B_i' = 0$, $i \in [q, q-1+M']$, $B_i' = B_i$, $i \in [q+M', M]$. Let an algorithm packs a dominate set of weights $D(K) = \{s(a_{I_1}), s(a_{I_2}), \dots, s(a_{I_K})\}$ into $B'$ bins and a number $K$ is maximal. It follows any set $D'(K) = \{s(a_{J_1}), s(a_{J_2}), \dots, s(a_{J_K})\}$ of $K$ weights that we can put into $B'$ bins will be dominated by $D(K)$: $s(a_{I_k}) \ge s(a_{J_k}), k \in [1, K]$. Then $n - K$ will be a maximal number of weights that we can put into the bins $B_i$, $i \in [q, Q - M']$. If we get a result $n - K < (i+1)(Q - q + 1 - M')$ for all $M' \in [1, \mu(i-1)]$ then we can remove the dominate group $G$ from $A$ and put $G$ into $B_Q$, after we remove $G$ and $B_Q$ from the initial $A$ and $L$ and set $A' := A \setminus G$, $L' := L \setminus B_Q$.

**Algorithm B3.** We define a set of numbers $J = \{1, 2, \dots n\} \setminus \{I_1, I_2, \dots, I_K\}$ that we can use as the numbers for the easiest weights. We denote $S_1(k) = \sum_{j=N_1(i)}^{N_1(i)+k-1} s(a_j)$ as a sum of $k$ heaviest weights from a range $[N_1(i), N_1(i) + M'' - 1]$, $S_2(k') = \sum_{j=n-k'(i+1)+k+1}^{n} s(a_{I_j})$ as a sum of $ik'$ easiest weights and $S_3(k') = \sum_{i=q}^{q+k'-1} B_i$ as a sum of $k'$ heaviest bins of range $[q, Q]$, $k' \in [1, k]$. If we get a result $S_1(k) + S_2(k') > S_3(k')$ for all $k' \in [1, k]$ for a fixed $k \in [1, M'']$, then we can remove $G$ from $A$ and put into $B_Q$, after we set $A' := A \setminus G$ and $L' := L \setminus B_Q$.

**Algorithm B(par).**
**1.** $A' := A$, $L' := L$.
**2.** $q := 0$, $G := \emptyset$.
**3.** $q := q + 1$. **If** ( $q > M$ ) **return 1**.

To build $N_1(i)$ for the $B := B_Q$, $i = 1, 2, \ldots H(B)$.
**If** ( $P(q) = P(q-1)$ ) **Go to 3**.
**4.** $Q := q - 1$.
**5.** $Q := Q + 1$. **If** ( $Q > M$ ) **Go to 3**.
**If** ( $B_Q = B_{Q+1}$ ) **Go to 5**.
**6.** $i := 0$, $\mu(0) := 0$.
**7.** $i := i + 1$.
**8.** $M' := Q - q + 1 - \mu(i-1)$. **If** ($M' = 0$ ) **Go to 5**.
**9.** Algorithms **B2** and **B3** to find $G$.
**10. If** ($G \neq \emptyset$ ) {
**If** ( par = 1 ) { **If** ( $sum(G) < \min_{q \le j \le Q} B_j^{min}$ ) **return 0** else **Go to 11** }.
Build $A' := A' \setminus G$ and $L' := L' \setminus B_Q$.
**If** ( $P(A', L')$=NO ) **return 0** else **Go to 2**. }
**11.** Algorithm **A1(**$B := B_q$**)** to find $\mu(i)$.
**12. Go to 7**.

The second block estimates an existence of reasonable solutions for a fixed number ($M$) of subsets. This block solves a problem: does exist a packing $A$ into $M$ bins: $0 < B_i^{\min} \le C_i \le B_i^{\max} \le B_i, i \in [1, M]$? We define a predicate $P(A, B^{\min}, B^{\max})$ =**NO**, if we claim "packing $A$ into $L$ doesn't exist" and $P(A, B^{\min}, B^{\max})$ =**YES** otherwise. A result of solving it problem is an estimate corridor $[C_i^{\min}, C_i^{\max}]$ : $B_i^{\min} \le C_i^{\min} \le C_i \le C_i^{\max} \le B_i^{\max}$, $i \in [1, M]$ that any reasonable solution $\{C_i\}$ will pass within $[C_i^{\min}, C_i^{\max}]$, $C_i^{\min} \le C_i \le C_i^{\max}$, $i \in [1, M]$, $C_i \ge C_{i+1}$. We denote $\lambda(h, H)$ as a maximal number of disjoint subsets that one can get from the initial $A$ in order to a sum of weights in each subset would belong to a range $[h, B]$. As a problem of finding of $\lambda(h, H)$ is NP-hard in the strong sense, we will find an upper bound $\nu(h, H) \ge \lambda(h, H)$. Below we give a recursive algorithm **A4** to build $\nu(h, H)$.

**Algorithm A4**
1. $A' := A, A^+ := \emptyset, s := 0, z_0 := 0, \nu(h, H) := 0$.
2. **For** $x = h$ **To** $H$
3. y:=0
4. **For** $k = 1$ **To** $n$
5. **If** ( $\exists A'' \subseteq A'$: $h \le \sum_{a_j \in A''} s(a_j) + s(a_k) \le x$ )
6. { $y := y + s(a_k)$, $s := s + s(a_k)$, $A' := A' \setminus a_k$, $A^+ := A^+ \bigcup a_k$ }.
7. **End**
8. $\lambda := \lfloor y/x \rfloor$.
9. **While** ( $\lambda > 0$ )
10. **If** ( $P(H, x, s, \lambda) = 0$ ) { $\lambda := \lambda - 1$ } else **Break** While.
11. **End** While
12. $\nu(h, H) := \nu(h, H) + \lambda, z_x := \lambda$.
13. **End**
14. **STOP**

**Algorithm** $P(H, x, s, \lambda)$
1. $K := \nu(h, H) + \lambda$, $A := A^+, M := K + 1$.

2. $B_i^{max} := x,\ B_i^{min} := h,\ i = 1, 2, \dots K,$
   $B_{K+1}^{max} := s - \lambda x - \sum_{i=1}^{x-1} z_i i,\ B_{K+1}^{min} := max\{s - Kx, 0\}.$
3. **If** ( **Algorithm B(0)** $= 0$ ) **return** 0.
4. **If** ( **Algorithm B(1)** $= 0$ ) **return** 0.
5. **return** 1.

Now we define:
An operator $P^+(h, H, x) = W^+ = \{w_i^+ \circ k_i^+\}$, where $w_i^+ = H - i + 1$,
$k_i^+ = \nu(H - i + 1, H) - \nu(H - i + 2, H),\ i \in [1, p],\ k_p^+ = x - \nu(H - p + 2, H)$,
$k_p^+ < \nu(H - p + 1, H) - \nu(H - p + 2, H),\ \sum_{i=1}^{p} k_i^+ = x,\ \nu(H + 1, H) := 0$,
a sum $S^+(h, H, x) = \sum_{i=1}^{p} k_i^+ w_i^+,\ w_p^+ < h \Rightarrow S^+(h, H, x) := 0,\ C^+ = \{C_j^+\}$ as
$C_j^+ = H, \qquad j \in \left[1, k_1^+\right],$
$C_j^+ = H - 1, \qquad j \in \left[k_1^+ + 1, k_1^+ + k_2^+\right], \dots$
$C_j^+ = H - p + 1,\ j \in \left[\sum_{i=1}^{p-1} k_i^+ + 1, \sum_{i=1}^{p} k_i^+\right].$
An operator $P^-(h, H, x) = W^- = \{w_i^- \circ k_i^-\}$ , where $w_i^- = h + i - 1$,
$k_i^+ = \nu(h + i - 1, h) - \nu(h + i - 2, h),\ i \in [1, p],\ k_p^- := x - \nu(h + p - 2, h)$,
$k_p^- < \nu(h + p - 1, h) - \nu(h + p - 2, h),\ \sum_{i=1}^{p} k_i^- = x,\ \nu(h - 1, h) := 0$,
a sum $S^-(h, H, x) = \sum_{i=1}^{p} k_i^- w_i^-,\ w_p^- > H \Rightarrow S^-(h, H, x) := \infty,\ C^- = \{C_j^-\}$ as
$C_j^- = h + p - 1,\ \ j \in \left[1, k_1^-\right],$
$C_j^- = h + p - 2,\ \ j \in \left[k_1^- + 1, k_1^- + k_2^-\right], \dots$
$C_j^- = h, \qquad j \in \left[\sum_{i=1}^{p-1} k_i^- + 1, \sum_{i=1}^{p} k_i^-\right].$

**Algorithm A5** to build the corridor $[C^{min}, C^{max}]$.
1. $C_i^{min} := B_i^{min},\ C_i^{max} := B_i^{max},\ i = 1, 2, \dots M$
2. $i := 0,\ REP := 0.$
3. $i := i + 1$. **If** ( $i > M$ ) **Go to** 6. $g := C_i^{max}$.
4. $C_i^{max} := \max_h \{\ C_i^{min} \le h \le min(g, C_{i-1}^{max}) : C_j^- \le C_j^{max},\ j = 1, 2, \dots i - 1,$
   $\sum_{j=1}^{i} C_j^- + \sum_{j=i+1}^{M} C_j^{min} \le S(A)\ \}$,
   where $C^- = \{C_1^-, C_2^-, \dots C_i^-\} = P^-(C_i^{max}, C_1^{max}, i)$.
5. **If** ( $C_i^{max} < g$ ) $REP := 1$. **Go to** 3.
6. $i := M + 1$.
7. $i := i - 1$. **If** ( $i < 1$ ) { **If** ( $REP = 1$ ) **Go to** 2 else **STOP** }. $g := C_i^{min}$.
8. $C_i^{min} := \min_h \{\ C_i^{min} \le h \le C_i^{max}\} : C_{i+j}^+ \ge C_{i+j}^{min},\ j = 1, 2, \dots M - i,$
   $\sum_{j=1}^{i-1} C_i^{max} + \sum_{j=1}^{M-i+1} C_i^+ \ge S(A)\ \}$,
   where $C^+ = \{C_1^+, C_2^+, \dots C_{M-i+1}^+\} = P^+(C_M^{min}, C_i^{min}, M - i + 1)$.
9. **If** ( $C_i^{min} > g$ ) $REP := 1$. **Go to** 7.

Now we show how to use our corridor. For Bin Packing we fix an initial $M = \lceil S(A)/B \rceil$ , define $B_i^{max} = B$, $B_i^{min} = \max\{s(a_n), S(A) - \sum_{i=1}^{M-1} C_i^{max}\}$ and form the corridor $[C_i^{\min}, C_i^{\max}]$. If we get $\sum_{i=1}^{M} C_i^{\max} < S(A)$ or $\sum_{i=1}^{M} C_i^{\min} > S(A)$, then we claim $P(A, B^{\min}, B^{\max})$ =**NO**, because of we increase $M$ by 1, form the corridor for new $M$ and so on. Last $M$ we take as final lower bound $M^1$. Using this technology to the reduced $A'$ we get finally $LB(A) = M^0 + M^1$. For the other models we build a bound on a following schema. At first we define a list $L$ using a features of models.

Then we lead $(A, L)$ to $(A', L')$ and set $A := A'$, $L := L'$. Further we define an initial corridor $B_i^{\min}$ and $B_i^{\max}$ and build an estimation corridor $[C_i^{\min}, C_i^{\max}]$. In case $P(A, B^{\min}, B^{\max})$ =**NO** we set other list $L$ and so on. A consecution of building the lists $L$ depend on the models.

On a base our estimation approach we developed new fast approximation algorithm **FG** [6] to solve **Model 1**. Here we give a description of algorithm **FG**.

**Algorithm FG**

1. Find $A^0$ and $M_0$ by algorithm **A**. Set $A'' := A \setminus A^0$.
2. Set the initial $a := \lceil S(A'')/B \rceil$ and $b := FFD(A'')$.
3. **While** ( $b > a$ )
4. Set $A := A''$, $M := a + (b - a)/2$, $L := \{B_i\}$, $B_i = B, i = 1, 2, \ldots M$.
5. **If** ( **B(0)** $= 0$ ) { Set $a := M + 1$; **Continue** While }
6. Lead $(A, L) \to (A', L')$ during **B**, set $A := A'$, $L := L'$, $n := |A'|$, $M' := |L'|$.
7. Find $k_0 = \max\limits_i \{ s(a_i) + s(a_{i+1}) + s(a_n) > B \}$, set $B_{M'-i+1} := B - s(a_i)$, $i = 1, 2, \ldots k_0$.
8. **For** $k = k_0 + 1$ **To** $n$
9. **For** $i = M'$ **To** 1 **By** -1
10. Set $gap := B_i - s(a_k)$. **If** ( $gap < s(a_n)$ ) **continue** For $i$
11. **If** ( $B_i < B$ ) { Find a maximal $s(a_p) \le gap$,
13. Set $A := A \setminus \{a_k \bigcup a_p\}, L := L \setminus B_i$ }.
14. **else** { Set $B_i := B - s(a_k)$, $A := A \setminus a_k$ }.
15. Sort $L$ by decreasing: $B_i \ge B_{i+1}$, **Break** For i (continue For $k$)
16. **End** For $i$
17. **If** ( $B_i - s(a_k) < s(a_n)$ ) for all $i \in [1, M']$ { Set $a := M + 1$, **continue** While}.
18. **End** For $k$
19. Set $b := M$, **continue** While.
20. **End** While
21. Set $FG(A) := M_0 + b$. **STOP**.

This algorithm shows the near-optimal results $p = 0.1 - 0.2\%$ in average for the largest data $m = n$ and range of weights $s(a_k) \in (0.25B; 0.5B]$. Here our initial reduction shows $|A^0| = n/3$. Thus, for this range we have the near-optimal lower and upper bounds. Using the techniques of our estimation approach we will develop the approximation algorithms for the other models of our list. Each such algorithm will be use both an initial reduction and estimation corridor by using the model properties.

A present program is written in Microsoft Visual C++. But we will develop our new product for the mass users to use in the Internet. With that purpose, we will use such modern program tools as C#2008, ASP.NET 3.5 and SQL Server 2005 to create a simple and mobile online tool for any people without special mathematical knowledge. Our estimation technology is universal one: this can be used to construct the algorithms to find the bounds of objective functions for the other problem tasks. Thus, our program tool is open to add new models. For a chosen model of our list a user can give a time limit. At first our program has to find a lower (upper) bound of objective function within a given time interval. Further the program will offer (by user'wish) an

approximation solution. If our bound is equaled to the approximation solution we state an optimal solution and finish our process. A productivity of program is controlled by set of parameters influencing to the runtimes of both bound and approximation solution. Thus, a main problem is to receive an approximation solution and measure a quality by using our bound. A user can compare own approximation solution using our bound. We observe too, there are a well-known commercial tool ILOG CPLEX to solve the combinatorial problems. But firstly, a tool price is very expensive to buy. Secondly, a user must have a special knowledge to transform own task to an input of CPLEX. Our future tool has a purpose to give to people an opportunity to solve the own tasks within a given time limit for the largest parameters $m$ (50000 and more).

## R E F E R E N C E S

1. Martello S., Toth P. Knapsack Problems. *John Wiley&Sons, Chichester*, 1990.

2. Coffman J.E.G., Galambos G., Martello S., Vigo D. Bin packing approximation algorithms: combinatorial analysis. D.-Z. Du, P. M. Pardalos (eds.). *Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Boston, MA.*, 1998

3. Applegate D.L., Buriol L.S., Dillard B.L., Johnson D.S., Shor P.W. The Cutting-Stock approach to bin packing: Theory and Experiments. *In Proceedings of the Fifth Workshop on Algorithm Engineering and Experimentation, R.E. Ladner (Editor), SIAM*, 2003, 1-15.

4. Fukunaga A.S., Korf R.E. Bin packing algorithms for multicontainer packing, knapsack, and covering problems. *Journal of Artificial Intelligence Research*, **28** (2007), 393-429.

5. Fedulov G. One-dimensional bin packing class: fast algorithms to find the bounds of objective functions. *Georgian Engineering News*, 2008, No. 2, 42-47.

6. Fedulov G. One-dimensional bin packing class: fast algorithms of finding the bounds of objective functions. *Sem. I.Vekua Inst. Appl. Math., Rep.,* **34** (2008), 78-88.

Received 13.05.2009; revised 15.06.2009; accepted 6.07.2009.

Authors' addresses:

G. Fedulov and N. Iashvili
Institute of Analytical Technique
36, Kakheti Highway, Tbilisi 0190
Georgia
E-mail: fedulov@caucasus.net
nugzar.iashvili @rambler.ru

Kh. Rukhaia and L. Tibua
I. Vekua Institute of Applied Mathematics of
Iv. Javakhishvili Tbilisi State University
2, University St., Tbilisi 0186
Georgia
E-mail: Khimuri.rukhaia@viam.sci.tsu.ge

K. Gulua
Sokhumi State University
9, Anna Politkovskaia St., Tbilisi 0186
Georgia
E-mail: gulua_x@mail.ru