An Optimized Multirobot Task Allocation

B. B. Choudhury and B. B. Biswal
Department of Mechanical Engineering, National Institute of Technology Rourkela, India
bbcnit@gmail.com, bibhuti.biswal@gmail.com

Abstract

Multirobot systems (MRS) hold the promise of improved performance and increased fault tolerance for large-scale problems. One of the most important aspects in the design of MRS is the allocation of tasks among the robots in a productive and efficient manner. Optimal solutions to multirobot task allocation (MRTA) can be found through an exhaustive search. Since there are $n \times m$ ways in which m tasks can be assigned to n robots, an exhaustive search is often not possible. Task allocation methodologies must ensure that not only the global mission is achieved, but also the tasks are well distributed among the robots. This paper presents task allocation methodologies for MRS by considering their capability in terms of time and space. A two-phase solution methodology is used to solve the MRTA problem wherein the task capacity of the robots is determined during the first phase and the task allocation optimization is done during the second phase using linear programming (LP).

1. Introduction

The study of multirobot system (MRS) has received increased attention in the recent years. Continually improving technology has made the deployment of MRS consisting of larger number of robots possible. Potential advantages of MRS over single robot systems (SRS) include reduction of total system cost by employing multiple simple and cheap robots as opposed to a single, complex and expensive robots. The inherent complexity of certain task environment may require the use of multiple robots as the demand for capability is quite substantial to be met by a single robot. Multiple robots are assumed to increase system robustness by taking advantage of inherent parallelism and redundancy.

The cooperation of robots in a group can be classified into two categories of implicit cooperation and explicit cooperation. In the implicit cooperation case each robot performs individual tasks, while the collection of these tasks is toward a unified mission. The explicit cooperation is the case where robots in a team work synchronously with respect to time or space in order to achieve a goal. In this paper, an attempt is made to empirically derive some guidelines for selecting task allocation strategies for multirobot systems with implicit cooperation. MRTA, in practice is a complex problem by itself and the formulation of this problem with multiple of robots of different types to take up large number tasks consists of several parameters that make it NP-hard. Therefore, the solution to this problem becomes computationally intractable. Heuristic approaches need to be followed to obtain a solution to such problems. However for finite and limited number of tasks it is possible as to the problem becomes NP-complete and optimization tools can be applied to get the solution. The present work aims at developing an approach for solving the MRTA problem considering a reduced domain. A generalized problem is formulated and considering the robots under question in terms of their space and time capabilities and the requirement of tasks an initial solution is obtained on the number of tasks that can be allocated to the candidate robots. Thereafter, LP technique is used to obtain the optimized MRTA. The approach presented in this paper can be advantageously used in real-world problems.

The allocation model (AM) is equivalent to a two-dimensional multi-type bin packing problem (2DMBP). Considering each robot as a two-dimensional bin, and each task a two-dimensional object to be packed, the model can be viewed as assigning objects into an optimal set of bins such that both resource demands of each object are satisfied and neither of the capacity constraints of each selected bin is violated. Historical research in bin packing has focused on both the one-dimensional single-type bin packing problem (1DSBP) and its two dimensional extension—2DSBP [1, 2]. Both 1DSBP and 2DSBP have been proven NP-hard [3,4]. Numerous investigators have examined the performance analysis of approximation algorithms designed for a number of

978-0-7695-3267-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ICETET.2008.111

2DSBP variants [5-10]. Reported algorithms may be categorized as either on-line or off-line. On-line algorithms pack a collection of objects that must be loaded as they are presented; in contrast, off-line algorithms pack the complete, possibly preordered, set of objects. Both types of algorithms are practical in real world applications. In addition, algorithms may also be categorized by the way that objects are packed. Specifically, the two packing strategies are box-packing and vector-packing. In box-packing, objects are treated as rectangles which are packed into an open-ended bin of unit width and infinite height. The objective in box-packing is to minimize the height of the open-ended bin. By contrast, in vector-packing, each object and bin is treated as a two-dimensional vector. The objective of vector-packing is to minimize the use of bins such that all objects packed into a bin have a component-wise sum less than or equal to one. To date, there exists no efficient 2DSBP optimization algorithm for solving problems of a practical size. To our knowledge, there also does not exist an exact optimization algorithm for 2DMBP. In this paper, the focus is on the development and implementation of an optimization algorithm that can be used to solve problems of a practical size within acceptable computational times.

The characteristics of AM warrant the development of an off-line algorithm involving vector-packing. Although the algorithm described here is in the context of robotics, it is general and applicable to any real-world application that requires the assignment of a set of two-dimensional objects to bins with two capacity constraints.

2. Allocation model formulation

The present research problem explicitly addresses robots of different types with various service time and space capacities. The allocation model (AM) seeks an optimal selection of robots to serve all given tasks such that each task's resource demands are satisfied, no robot capacity constraints are violated, and the total system cost is minimized. A heuristic model along with its solution algorithm is presented for allocation of robots to the tasks which is efficient and may serve as a planning tool. The initial model is formulated as a pure 0-1 mathematical program. The two most important factors when assigning tasks to robots are the geometrical work envelope and the kinematic machine cycle time. The work envelope for a typical robot is represented by a diameter of a circle. However, for our model, it is not required that the work envelope be a complete circle. The time requirement of any task depends upon its relative distance from the robot. Thus, there exists a trade-off between the space requirement and machine cycle time requirement. In fact, both requirements are a function of the task's relative position from the robot. Our primary objective is to minimize the total robot allocation costs while satisfying task resource demands. In order to make AM computationally tractable, it is assumed that all tasks are placed at the most remote location within the work envelope. This assumption decouples the interaction between space and time by allowing the resource requirements of a given task to be constant. Without this assumption, the model complexity is significantly increased. This trade-off between the number of robots required to serve a given set of tasks and the time required to serve a task could be considered by iteratively solving AM. The formulation of the AM is as follows. A set of robot types indexed by $K = \{1, 2, \ldots, k\}$, is considered where each robot type is characterized by its time and space capacity. Specifically, space is measured in terms of the work envelope's swept area. All given tasks are indexed by $I = \{1, 2, \ldots, n\}$. Each task i demand a known amount of time and space when served by robot type k, denoted by t_{ik} and s_{ik} respectively.

Table 1. Notation for AM

Notation	Definition
k	Number of robot types
K	Robot type index set, K = {1, 2, ..., k}
n	Number of tasks
I	Task index set, I = {1, 2, ..., n}
m_k	Maximum number of type k robots $\forall k \in K$
t_{ik}	Normalized time requirement of task i when served by a type k robot, $\forall i \in I, k \in K$
s_{ik}	Normalized space requirement of task i when served by a type k robot, $\forall i \in I, k \in K$

In addition, for a given set of n tasks, let m_k denote the maximum number of robots of type k necessary to serve all tasks assuming only robots of type k are available. Further, let K denote the index set for type k robots. For easy reference, all useful notations for AM are summarized in Table 1. A decision variable, x_{ik}, is defined as:

$$x_{ik} = \begin{cases} 1 \; if \;\; workstation \; i \; \text{is assigned to robot of type } k \\ 0 \;\; \text{otherwise} \end{cases}$$

With no loss of generality, the time and space requirements for each task (i.e., t_{ik}, and s_{ik}, respectively) can be normalized by dividing the robot resource capacities into the corresponding task resource demands. The robot selection and assignment

(RSA) can be written as equation (1) through equation (5):

$$(\text{RSA})\ \text{MIN}\ Z = \left(\sum_{k \in K} f_k \left(\sum_{k \in k} x_{ik} \right) \right) \quad (1)$$

Such that

$$\sum_{k \in K} \sum_{k \in k} x_{ik} = 1 \quad \forall i \in I \quad (2)$$

$$\sum_{i \in I} t_{ik} x_{ik} = 1 \quad \forall k \in K_k \quad (3)$$

$$\sum_{i \in I} s_{ik} x_{ik} \leq 1 \quad \forall k \in K \quad (4)$$

$$x_{ik} \in \{0,1\} \quad \forall i \in I, k \in K \quad (5)$$

Condition (2) ensures that each task *i* is assigned to exactly one robot. Conditions (3) and (4) ensure that tasks assigned to any robot will not violate the corresponding time and space constraints. The AM is a pure 0-1 integer program (IP). Therefore, it is impractical to directly solve AM by using any available IP code. In the present work an optimization algorithm based on heuristic covering all the necessary parameters is developed for solving the task assignment problem in a heterogeneous multirobot environment. This is a general two-dimensional problem that is NP-hard, and it is computationally intractable. The MRTA becomes computationally tractable by using a two-phase methodology. During the first phase, the RSA provides the initial solution for distribution of tasks which is then formulated as linear programming problem to be solved in the second phase.

3. Methodology

Before allocation of tasks to the candidate robots, it is essential to determine the capacity of the candidate robots. The task capacity of a robot depends on its reach, speed, and pay load specifications. In addition to the robot's capability, the loading capacity also depends on the requirement of the task (kind of operation, motion, and dimensions) and its location in the workspace. During the first phase, the task capacity is determined by using the load deviation ratio which uses the normalized time and space requirement for various combinations of robot-task. The load deviation ratio encompasses all the required parameters for deciding the loading capacity. Load deviation ratio is the ratio of difference between the normalized space requirement (s_{i1}) and normalized time requirement (t_{i1}) to the summation of normalized space requirement (s_{i1}) and normalized time requirement (t_{i1}).

However, task assignment in cooperative control requires online real-time solution. A single robot is able to service multiple tasks. Furthermore, some tasks must be serviced following a specific sequence in time. Therefore, task assignment for cooperative control is fundamentally different from off-line static task assignment studied in the literature. When a task is detected, it needs to be classified. If classified, the task can be assigned. An important point is once a task is assigned, the task is viewed by other robots to ensure it has been assigned. The tasks must be correctly assigned and distributed as per load deviation ratio. Therefore, the task assignment in cooperative control is a dynamical process with unexpected and changes in the task in the system, and in the environment. During the second phase, the problem of multiple task assignment using linear programming is formulated.

4. Algorithm details

4.1. Allocation heuristic (AH)

The task may be time intensive, space intensive or neither. A heuristic is developed by examining these three cases and the load balance on each candidate robot. The AH is based on the concept of allocation cost, which is computed as a function of the resource demands of each task and a robot's load balance. Let Δ_k denote the load balance factor associated with the robot of type *k*. That is, Δ_k is defined as the difference between the total allocated (normalized) machine time and the total allocated (normalized) work space for robot of type *k*. Hence, Δ_k can be expressed as follows:

$$\Delta_k = T_k - S_k \quad \forall k \in K$$

If a robot's resource load is nearly, balanced, then the load balance factor will be approximately zero. If the robot's load is time intensive, then $0 < \Delta_k < 1$, and if the robot's load is space intensive, then $-1 < \Delta_k < 0$. Hence, the further the resource load factor is away from zero, greater the load imbalance is. In addition, let δ_{ik} denote the adjusted demand when the i^{th} task is served by the robot of type *k*. That is,

$$\delta_{ik} = \begin{cases} \text{MAX}\{t_{ik}, s_{ik} - \Delta_k\} \text{ if } \Delta_k > 0 \\ \text{MAX}\{t_{ik} + \Delta_k, s_{ik}\} \text{ if } \Delta_k \leq 0 \end{cases} \quad (6)$$

Since $0 < t_{ik} \leq 1$, $0 < s_{ik} \leq 1$, and $-1 < \Delta_k < 1$, we know $0 < \delta_{ik} \leq 1$. To illustrate how the adjusted demand is employed by AH, consider two robots of type *k*, say A and B. Assume that $\Delta_{Ak} = 0.4$ and $\Delta_{Bk} = -3$.

Therefore, robot A is time intensive. To improve the load balance for robot A, the assignment of a task which is space intensive (i.e., $t_{ik} < s_{ik}$) should be preferred to those which are time intensive. By contrast, for robot B, the assignment of tasks which are time intensive should be given preference over tasks which are space intensive. As an example, let the task to be assigned next is time intensive; that is, t_{ik}= 0.3 and s_{ik}= 0.2 and assume both robot A and B have enough remaining time and space capacities to serve this candidate task. The goal of the assignment heuristic is to balance the resource load on each robot. Since the candidate task is time intensive, it should be assigned to a robot which is space intensive. Plugging the given figures into equation (7), we have $\delta_{iAk} = 0.3$ and $\delta_{iBk} = 0.2$. These adjusted demands, i.e., δ_{ik} contribute to the "allocation costs". In general, if the fixed cost of all robot types is equal, the task should be assigned to the robot which produces the smallest adjusted demand. Since, not all robots have equal fixed cost, the allocation cost, a_{ik} incurred by the i^{th} task when served by the robot of type k is the product of its adjusted demand and the fixed cost of the robot, i.e.,

$$a_{ik} = f_k * \delta_{ik} \qquad (7)$$

Since $0 < \delta_{ik} \leq 1$, we have $0 < a_{ik} \leq f_k$. Thus, a_{ik} reflects the adjusted proportion of the fixed cost that task i incurs when it is assigned to robot of type k. For each robot type k, the heuristic calculates the load deviation ratios and sorts them into a nondecreasing order. These load deviation ratios indicate the balance between the time and space requirements of each task when served by each robot type k. Then, the AH is employed to assign tasks to robots based on the sorted load deviation ratios.

Table 2. Fixed costs and parameter values of the robots

	Robot-1	Robot-2	Robot-3	Robot-4
Specification	(Puma 560-c)	(Adept one XL)	Fanuc Arcmate Sr.R.J	Staubli RX 130B
DOF	6	4	6	6
Pay Load	4 kg	12 kg	10 kg	12 kg
Swept Area	320°	270°	300°	320°
Max. Reach	878 mm	800 mm	1529 mm	1250 mm
Max Speed	1.0 m/sec	1.2 m/sec	3.60 m/sec	3.09 m/sec
Type	Jointed	Scara	Jointed	Jointed
Cost	$35,000	$19,500	$56,400	$60,000

Table 3. Normalized space and time requirement of tasks

Task		Normalized space requirement				Normalized time requirement			
No. (i)	Size (D)	R-1	R-2	R-3	R-4	R-1	R-2	R-3	R-4
		S_{i1}	S_{i2}	S_{i3}	S_{i4}	t_{i1}	t_{i2}	t_{i3}	t_{i4}
1	1.0	0.216	0.286	0.127	0.147	0.214	0.216	0.203	0.2
2	0.7	0.146	0.192	0.088	0.101	0.143	0.145	0.141	0.142
3	1.1	0.242	0.321	0.14	0.163	0.237	0.243	0.225	0.228
4	1.05	0.229	0.303	0.133	0.155	0.224	0.229	0.213	0.216
5	0.9	0.192	0.253	0.114	0.131	0.188	0.191	0.181	0.184
6	1.01	0.219	0.289	0.128	0.148	0.215	0.219	0.205	0.208
7	0.65	0.135	0.177	0.081	0.094	0.133	0.134	0.13	0.131
8	0.7	0.146	0.192	0.088	0.101	0.143	0.145	0.14	0.142
9	0.75	0.158	0.207	0.094	0.109	0.154	0.156	0.15	0.152
10	0.85	0.18	0.237	0.107	0.124	0.177	0.179	0.171	0.173
11	1.1	0.242	0.321	0.14	0.163	0.237	0.243	0.224	0.227
12	1.5	0.366	0.515	0.195	0.23	0.359	0.39	0.313	0.322
13	1.4	0.33	0.452	0.181	0.212	0.324	0.342	0.29	0.297
14	1.2	0.269	0.359	0.154	0.179	0.264	0.272	0.246	0.25
15	1.18	0.263	0.351	0.151	0.176	0.258	0.266	0.241	0.245

4.2. Example problem

Using realistic data, the following example is provided to highlight the solution process for an AM problem. Table 2 summarizes major parameter values for four different robot types, and Table 3 presents the normalized space and time requirements of fifteen tasks. Robot-4 has a fixed charge of $60,000, a swept area of 320°, a maximum reach of 1250 mm, and an average arm movement speed of 3.09 m/sec. Each entry in column two of Table 3 provides the diameter (D) of a circle encompassing the task. It is assumed that each task is placed at the most remote location within the work envelope. Therefore, the D associated with each task is in fact a chord to the work envelope. Knowing the value of D and the maximum reach (R) of a robot, the arc length subtended by a task is $R\theta$ *where* $\theta = 2\sin^{-1}(D/2R)$. Here, θ represents the task's space requirement in degrees. Considering task one and robot type one, we have D = 1.0 meter, R = 1.25 meters, and S = 320°. Using these data, θ =47.15° and $S_{11} = (47.15/320) = 0.147$.

In this model, the time requirement for each task is estimated based on two major components: robot arm travel time, and robot service time. Both components are normalized by the total available machine time, which in practice is defined by the time available during peak machine hours. To proceed with the solution for allocation model, all the options of employing individual and/or combination of available robot types are tried. Table 4 provides the load balance factors calculated for the robots. This is a problem specific condition and it largely depends on number of

factors such as time and space requirement. This is mainly due to low value of task size and relatively high value of the speed of the robots. Table 5 presents the allocation cost of four robots. The load balance factor, time requirement, space requirement and allocation cost are considered for the assignment of the robots to the tasks in question. Figure 1 shows the capacity curves of the four individual robots that decides the allocation strategy.

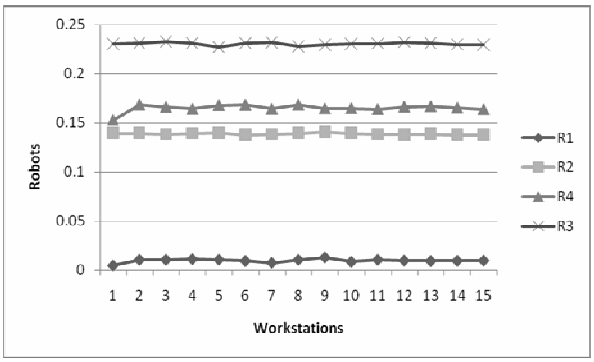

Figure 1. Load deviation ratio of robots

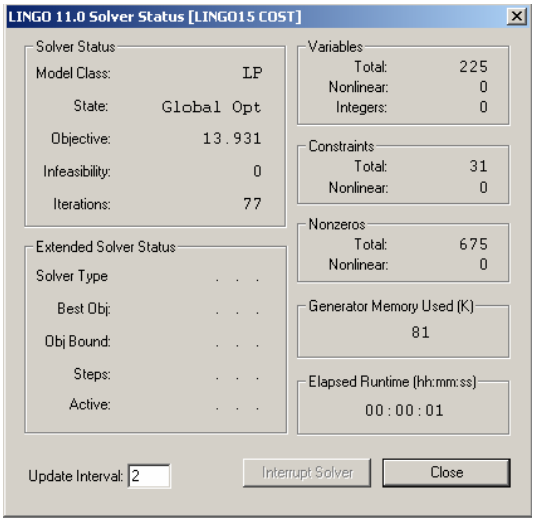

Figure 2. Results of the LP using LINGO

LINGO is a comprehensive tool designed to make building and solving linear, nonlinear and integer optimization models faster, easier and more efficient. It provides a completely integrated package that includes a powerful language for expressing optimization models, a full featured environment for building and editing problems, and a set of fast built-in solvers. For creating a LINGO model, an optimization model consists of three parts; a) objective function – that describes exactly what the model should optimize, b) variables – the quantities that can be changed to produce the optimal value of the objective function, and c) constraints – the formulae that define the limits on the values of the variables.

Table 4. Load deviation ratio

Task	R-1	R-2	R-3	R-4
	LDR	LDR	LDR	LDR
1	0.004	0.139	0.23	0.153
2	0.01	0.139	0.231	0.169
3	0.01	0.138	0.232	0.166
4	0.011	0.139	0.231	0.164
5	0.01	0.139	0.227	0.168
6	0.009	0.137	0.231	0.169
7	0.007	0.138	0.232	0.164
8	0.01	0.139	0.228	0.169
9	0.012	0.14	0.229	0.165
10	0.008	0.139	0.23	0.165
11	0.01	0.138	0.23	0.164
12	0.009	0.138	0.232	0.167
13	0.009	0.138	0.231	0.167
14	0.009	0.137	0.23	0.166
15	0.009	0.137	0.229	0.164

Table 5. Allocation cost of assigned task

W.S	R1	R2	R4	R3
1	0.475	0.355	0.786	0.733
2	0.513	0.384	0.852	0.795
3	0.513	0.384	0.852	0.79
4	0.553	0.414	0.912	0.846
5	0.592	0.444	0.972	0.902
6	0.633	0.475	1.038	0.964
7	0.658	0.5	1.086	1.015
8	0.674	0.507	1.104	1.021
9	0.699	0.526	1.14	1.06
10	0.759	0.573	1.2	1.145
11	0.767	0.579	1.248	1.156
12	0.803	0.607	1.296	1.201
13	0.848	0.642	1.368	1.269
14	0.848	0.643	1.362	1.263
15	0.923	0.703	1.47	1.359

The Solver Status box as shown in Figure 2 details the model classification (LP, QP, ILP, IQP, NLP, etc.), state of the current solution (local or global optimum, feasible or infeasible, etc.), the value of the objective function, the infeasibility of the model (amount constraints are violated by), and the number of iterations required to solve the model. After the solver status box the LINGO displays a solution report regarding the values of each variable and the complete allocation that will produce the optimal value of the objective function. The reduced cost for any variable that is included in the optimal solution is always zero. For variables not included in the optimal solution, the

reduced cost shows how much the value of the objective function would decrease (for a MAX problem) or increase (for a MIN problem) if one unit of that variable were to be included in the solution.

5. Results and analysis

The optimization algorithm discussed in the previous section was coded in LINGO for solving the linear program. All test problems are created by a problem generator using four major design parameters: 1) the average robot service capacity (i.e., the average number of tasks that can be served by a robot based on one-dimensional resource demand of tasks), 2) the average space required by the given tasks, 3) the average machine time required by the given tasks, and 4) the number of tasks to be assigned. For all test problems, four robot types with fixed charges are considered as candidate robots. The results of the allocation are presented in Table 6. The initial feasible solution generated by the heuristic takes no more than a second. The quality of the solution is reasonably good. The solution times for finding a near-optimum or an optimum are also recorded. Thus, the algorithm provides significant and useful results. The total cost of assigned task is 13.931.

Table 6. Robot selection and assigned task

Robot	Assigned task
Robot-1	WS-13
Robot-2	WS-12, WS-14, WS-15
Robot-3	WS-5 WS-6, WS-7, WS-8, WS-10, WS-11
Robot-4	WS-1, WS-2, WS-3, WS-4, WS-9

6. Conclusions

Multirobot facility design and planning have become increasingly important in modem production over the past decade. In this paper, a mathematical model and solution algorithm is developed to support robot selection and task assignment in a system employing multiple robot types. Specifically, our model considers selection of a proper mix of multiple-type robots such that operational requirements for a given number of tasks are satisfied. Each robot is characterized by its unique fixed charge and subject to its machine time and space capacity constraints. Each task has known time and space demands for each type of robot. A model of L.P is developed for multirobot assignment. The result implies that that the size of the linear programming is determined by the number of tasks, independent of robots linear. The model is initially formulated as a pure 0-1 mathematical program. The initial solution obtained from the first phase is utilized to decide the task performing capacities of the candidate robots. The model is then simulated by number of tasks to make it suitable for application of LP in order to find out the optimized task allocation. In order to test the efficiency of the methodology an example problem with four heterogeneous robots and fifteen different tasks is worked out. Computational results indicate that the algorithm is effective and efficient in solving problems of a practical size.

7. References

[1] K. A. Dowsland and W. B. Dowsland, "Packing problems", *European Journal of Operational Research*, vol. 56, 1992, pp. 2-14.

[2] D. S. Johnson, "Fast algorithms for bin packing", *Journal of Computers and System Sciences*, vol. 8, (1974), pp. 272-314.

[3] Aho, A. V, Hopcroft, J. E, and Ullman, J. D, "The Design and Analysis of Computer Algorithms". Reading, MA: Addison-Wesley, 1974, ch. 10.

[4] B. S. Baker, E. G. Coffman, and R. L. Rivest, "Orthogonal packings in two dimensions", *SIAM Journal of Computing*. vol. 9, no. 4, pp. 846855, 1980.

[5] B. S. Baker and J. S. Schwarz, "Shelf algorithms for two-dimensional packing problems", *SIAM Journal of Computing,* vol. 12, pp. 508-525, 1983.

[6] D. J. Brown, B. S. Baker, and H. P. Katseff, "Lower bounds for online two-dimensional packing algorithms", *Acta Informatica*, vol. 18, pp. 207-225, 1982.

[7] F. R. K. Chung, M. R. Garey, and D. S. Johnson, "On packing two-dimensional bins", *SIAM Journal on Algebraic and Discrere Methods,* vol. 3, pp. 66-76, 1982.

[8] B. Chazelle, "The bottom-left bin-packing heuristic: An efficient implementation", *IEEE Transactions on Computers*, vol. 32, no. 8, pp. 697-707, August 1983.

[9] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin-packing—An updated survey", *in Algorithm Design for Computer System Design*, Springer Verlag, pp. 49-106, 1984.

[10] W. T. Rhee and M. Talagrand, "Multidimensional optimal bin packing with items of random size", *Mathematics of Operations Research*, vol. 16, pp. 49C503, 1991.