AN EFFICIENT ALGORITHM FOR OPTIMAL ASSIGNMENT OF COMPONENTS AND BOARDS IN PRINTED CIRCUIT BOARD MANUFACTURING

PRATTANA MANEECHAY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF SCIENCE IN APPLIED MATHEMATICS
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2003

ISBN 974-324-760-2

เลขหมู่.....
เลขทะเบียน 41542
วัน,เดือน,ปี 30 ธ.ค. 2547
.b.....
.i.....
Copyright© by Chulalongkorn University
This thesis is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

COPYRIGHT 2003

SCHOOL OF GRADUATE STUDIES

This material is reserved for educational use only, not allowed for commercial use.

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

Forbidden to modify the content, and cite the document when use.

หัวข้อวิทยานิพนธ์	**ขั้นตอนวิธีที่มีประสิทธิภาพสำหรับจัดอุปกรณ์อิเล็กทรอนิกส์ที่เหมาะสมที่สุดในการผลิตแผงวงจรอิเล็กทรอนิกส์**
นักศึกษา	**นางสาวปรารถนา มณีฉาย**
รหัสประจำตัว	39065301
ปริญญา	**วิทยาศาสตรมหาบัณฑิต**
สาขาวิชา	**คณิตศาสตร์ประยุกต์**
พ.ศ.	2546
อาจารย์ผู้ควบคุมวิทยานิพนธ์	**รศ.อุบลวรรณา เงินวิจิตร**
อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม	**รศ.ดร.วีระ จันทร์ดง**

บทคัดย่อ

งานวิจัยนี้ศึกษาปัญหาการจัดชิ้นส่วนอุปกรณ์ในการผลิตแผงวงจรอิเล็กทรอนิกส์ (Printed Circuit Board) ซึ่งปัญหาจะอยู่ในรูปแบบกำหนดการจำนวนเต็มแบบทวิภาค (Binary Integer Program) การหาผลเฉลยใช้วิธี Variable Decomposition โดยอาศัยโครงสร้างพิเศษของปัญหา และพัฒนาขั้นตอนวิธีในการหาผลเฉลย เปรียบเทียบว่าวิธีนี้มีประสิทธิภาพดีกว่าวิธีเดิมที่มีอยู่ในเชิงผลเฉลยที่ได้ โดยใช้ปัญหาตัวอย่างที่ผู้วิจัยสร้างขึ้นเอง

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Thesis Title	An Efficient Algorithm for Optimal Assignment of Components and Boards in Printed Circuit Board Manufacturing
Student	Miss Prattana Maneechay
Student ID.	39065301
Degree	Master of Science
Programme	Applied Mathematics
Year	2003
Thesis Advisor	Assoc.Prof.Ubolwanna Ngernwichit
Thesis Co-advisor	Assoc.Prof.Dr.Vira Chankong

ABSTRACT

This research aims at studying the problem of operation assignment in the production of printed circuit boards. This problem can be expressed as a binary integer program (BIP). We propose the use Variable Decomposition method which will be specially customized to take advantage of the particular structure of the problem at hand. By examining the special structure of the problem and developing a custom solution procedure for it, we will show that it is much more efficient than any of the existing technique in terms of quality of solution. We will do so using several test problems, obtained from the self-generated.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and deep appreciation to my major advisor, Assoc.Prof.Ubolwanna Ngernwichit, for the valuable instruction, guidance, excellent encouragement and kindness to me throughout this study.

I am equally grateful to my co-advisor, Assoc.Prof.Dr.Vira Chankong, who has offered the topic of this thesis, for his helpful and continual suggestions, great supervision and constructive criticisms throughout this research which enable me to complete this thesis successfully.

I also thank to the Department of Mathematics and Computer Science, Faculty of Science, KMITL for providing a scholarship, which enable me to undertake this study.

Special thanks are extended to Mr.Phitoon Srinil , for developing the program. And my graduate friends for their friendly assistance and encouragement during preparing this thesis.

Finally, I would like to express my eternal gratitude to my parents for their endless support, understanding, love and care.

Prattana Maneechay

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

TABLE OF CONTENTS (continue)

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

LIST OF TABLES

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

LIST OF FIGTURES

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

CHAPTER 1

INTRODUCTION

1.1 STATEMENT AND SIGNIFICANCE OF THE PROBLEMS

We consider an operation assignment problem that arose from a printed circuit board (PCB) assembly process. Components can either be inserted on boards manually or by machine. The objective is to determine an assignment of components (operation) to a set of capacitated machines (with the remainder of the components inserted manually) to minimize the total set-up and processing cost for assembling all boards.

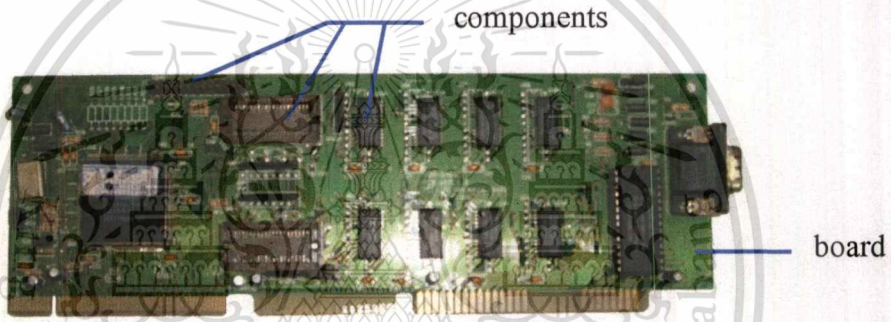

Figure 1.1 Components and board of printed circuit board

In 1991 M. L. Brandeau and C. A. Billington [7], they consider operation assignment problem. The problem can be formulated as a mixed integer linear program, but is too large to be practically solved. For the case of one machine, they present two different solution heuristics. To show that while each can be arbitrarily bad, on average the algorithms perform quite well. For the case of multiple machines, they present four different solution heuristics. Which consider the solution of problem with two heuristics as follow Stingy Component algorithm and Greedy Board algorithm, which both algorithms for solved operation assignment problem that arose from a PCB assembly process are simplicity, but solution is not optimal.

In 1998 M. S. Hillier and M. L. Brandeau [9], they consider operation assignment problem. An optimal solution technique is developed for the single-

Forbidden to modify the content, and cite the document when use.

machine case and for the multiple-machine case where boards are not allowed to be set-up on more than one process. Which solving problem is using to the linear programming (LP) by Lagrangian Relaxation. In addition, a heuristic is developed which gives near-optimal solutions (within 0.3%) with much less computational effort.

In this research, we propose the use Variable Decomposition method which will be specially customized to take advantage of the particular structure of the problem at hand. By examining the special structure of the problem and developing a custom solution procedure for it, we will show that it is much more efficient than any of the existing technique in terms of quality of solution and computational cost.

1.2 OBJECTIVE OF THE STUDY

The research developed the custom solution algorithm by Variable Decomposition method, which replace general purpose and heuristics.

1.3 SCOPE OF THE STUDY

This research aims at studying the binary integer program (BIP):

$$\text{Minimize} \quad \sum_i \sum_j \sum_k c_{ij} v_{jk} x_{ijk} + \sum_i \sum_k s_{ik} d_k y_{ik} \tag{1.1}$$

Subject to

$$\sum_i x_{ijk} = r_{jk} \qquad \forall j, k \tag{1.2}$$

$$y_{ik} \geq x_{ijk} \qquad \forall i, j, k \tag{1.3}$$

$$z_{ij} \geq x_{ijk} \qquad \forall i, j, k \tag{1.4}$$

$$\sum_j z_{ij} \leq N_i \qquad i = 1, 2, \ldots, I-1 \tag{1.5}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j, k \tag{1.6}$$

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

$$y_{ik} \in \{0,1\} \qquad \forall i, k \qquad (1.7)$$

$$z_{ij} \in \{0,1\} \qquad \forall i, j \qquad (1.8)$$

The solving operation assignment problem that arose from a PCB assembly process is using to the Variable Decomposition method. In this research developed algorithm for solve the solution. Compare algorithm and solution with Greedy Board Algorithm.

1.4 PROCESS OF THE STUDY

1. Research journals and information.
2. Developed algorithm for solving the problem.
3. Solving the operation assignment problem that arose from a PCB assembly process.
4. Compare the solution between Variable Decomposition method with Greedy Board algorithm.
5. Conclude and discuss the result.
6. Formulate a written thesis.

1.5 EXPECTED RESULTS

1. We received algorithm for solving the operation assignment problem that arose from a PCB assembly process.
2. To be basic in study for research.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 2

LITERATURE REVIEW

In chapter that we state about research which associate with solving the operation assignment problem that arose from a PCB assembly process. The problem is binary integer program (BIP).

BIP 1: Original Formulation

Minimize $$\sum_i \sum_j \sum_k c_{ij} v_{jk} x_{ijk} + \sum_i \sum_k s_{ik} d_k y_{ik} \quad (2.1)$$

Subject to

$$\sum_i x_{ijk} = r_{jk} \qquad \forall j, k \quad (2.1.1)$$

$$y_{ik} \geq x_{ijk} \qquad \forall i, j, k \quad (2.1.2)$$

$$z_{ij} \geq x_{ijk} \qquad \forall i, j, k \quad (2.1.3)$$

$$\sum_j z_{ij} \leq N_i \qquad i = 1, 2, \ldots, I-1 \quad (2.1.4)$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j, k \quad (2.1.5)$$

$$y_{ik} \in \{0, 1\} \qquad \forall i, k \quad (2.1.6)$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j \quad (2.1.7)$$

where

i = processes ($i = 1, \ldots, I$-1: machine; $i = I$; manual)

j = components ($j = 1, \ldots, J$)

k = boards ($k = 1, \ldots, K$)

c_{ij} = cost of insertion of component j on process i

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

s_{ik} = setup cost for board k on process i

d_k = expected demand for board k during the time horizon

$$r_{jk} = \begin{cases} 1 & \text{if component j is used in board k} \\ 0 & \text{otherwise} \end{cases}$$

n_{jk} = number of component j used in board k

v_{jk} = expected number of component j used in board k during the time horizon ($=d_k n_{jk}$)

N_i = total number of different types of components that can be assigned to process i

$$x_{ijk} = \begin{cases} 1 & \text{if component j of board k is assigned to process i} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{if board k is set up on process i} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if component j is assigned to process i} \\ 0 & \text{otherwise} \end{cases}$$

In 1991 M. L. Brandeau and C. A. Billington [7], they develope two heuristics approach for solving the operation assignment problem. They discuss implementation of their results at Hewlett-Packard.

In the first case, the single machine problem will divided to solve problem be two algorithms, Stingy Component algorithm and Greedy Board algorithm.

Algorithm 2.1 : Stingy Component Algorithm

1. Initialization: Let S = {1, ...,J}.

   Let $\delta_k = 1$ for all k, and $J = \sum_{k=1}^{K} (v_{jk} c_1 + s_1)$

   If $|S| \leq N_1$, STOP.

2. "Stingy" Component Removal:

   Calculate $\Delta_j = \sum_{k=1}^{K} [v_{jk}(c_2 - c_1) + r_{jk}\delta_k s_2]$

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

For all $j \in S$. Find $l = \arg\min_{j \in S} [\Delta_j]$

Let $S = S - l$, $J = J + \Delta_1$, and for k such that $\delta_k r_{lk} = 1$, set $\delta_k = 0$.

3. Post – Processing; If $|S| > N_1$, return to Step 2. Otherwise,

   (i) For all $j \notin S$: set $x_{2jk} = r_{jk}$, $x_{1jk} = 0$, for all k.

   (ii) For all k such that $\delta_k = 1$: set $y_{1k} = 1, y_{2k} = 0$ and $x_{2jk} = 0, x_{1jk} = r_{jk}$ for all j.

   (iii) For all k such that $\delta_k = 0$: set $y_{2k} = 1$, and

$$\text{if } s_1 + \sum_{j \in S} v_{jk} c_1 > \sum_{j \in S} v_{jk} c_2$$

then set $y_{1k} = 0, x_{1jk} = 0 \ \forall j, x_{2jk} = r_{jk} \ \forall j$ and

$$J = J + \left[\sum_{j \in S} v_{jk} (c_2 - c_1) \right] - s_1$$

otherwise, set $y_{1k} = 1$, and $x_{1jk} = r_{jk}$, , $x_{2jk} = 0 \ \forall j \in S$

Stop.

Step 1 assigns all components to the machine.

Step 2 the incremental cost of removing each component from the machine (Δ_j) is calculated; this consists of the incremental variable processing cost per unit ($c_2 - c_1$) times the total affected volume ($\sum_k v_{jk}$), plus the incremental manual set-up cost for any boards using that component which have no yet incurred a manual set-up cost (i.e. those boards for which $\delta_k = 1$ at this step). The component which adds the minimum incremental cost is removed, and the objective function, set-up indicators (δ_k), and costs are updated. The removal process continues until the bin capacity of the machine is exactly met.

Step 3 is a post-processing step that sets the decision variables. All boards which incur no manual set-up ($\delta_k = 1$) are processed completely by the machine. For those boards that do incur a manual set-up ($\delta_k = 0$), a decision is made about whether or not to process the board entirely on the manual process; if it is

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

cheaper to do so, the decision variables are adjusted and the objective function is updated.

Algorithm 2.2: Greedy Board Algorithm

1. Initialization: Let $S = \varnothing$, $T = \{1, \ldots, K\}$, and

$$J = \sum_{k=1}^{K} [v_{jk} c_2 + s_2]$$

2. "Greedy" Board Loading: Calculate

$$\gamma_k = \left[\sum_{j \notin S} r_{jk} \right] \Big/ d_k \quad \text{for all } k \in T'$$

where

$$T' = \left\{ k \in T : \sum_{j \notin S} r_{jk} \leq N_1 - |S| \right\}$$

Find $m = \arg\min_{k \in T'} [\gamma_k]$

Let $T = T - m$, $S = S + \{j : r_{jk} = 1\}$, $J = J + [v_{jk}(c_1 - c_2) + s_1 - s_2]$

3. Post-Processing: If $T' \neq \varnothing$, return to Step 2. Otherwise,

(i) For all $j \notin S$: set $x_{2jk} = r_{jk}$, $x_{1jk} = 0$, for all k.

(ii) For all $k \notin T$: set $y_{1k} = 1$, $y_{2k} = 0$, and $x_{2jk} = 0$, $x_{1jk} = r_{jk}$ for all j.

(iii) For all $k \in T$: set $y_{2k} = 1$, and

$$\text{If } s_1 + \sum_{j \in S} v_{jk} c_1 > \sum_{j \in S} v_{jk} c_2$$

Then set $y_{1k} = 1$, $x_{1jk} = r_{jk}$ $\forall j$, $x_{2jk} = 0$ $\forall j$ $y_{1k} = 1$, $x_{1jk} = r_{jk}$ $\forall j$, $x_{2jk} = 0$ $\forall j$, and

$$J = J + \left[\sum_{j \in S} v_{jk} (c_1 - c_2) \right] + s_1$$

otherwise, set $y_{1k} = 0$, and $x_{1jk} = 0$, $x_{2jk} = r_{jk}$ $\forall j \in S$

Stop.

Step 1 assigns all boards to the manual process.

Step 2 the incremental number of new component slots per board produced (γ_k) is calculated for each board whose incremental assignment to the machine will

This material is protected by copyright. Forbidden to modify the content, and cite the document when use.

not violate the slot capacity constraint (represented by $k \in T'$), and the board with the minimum value is assigned to the machine. This is equivalent to greedily maximizing incremental boards produced per additional slot used. The process continues until on more (entire) boards can be assigned to the machine.

Step 3, the post-processing step, is similar to that for the Stingy Component algorithm (in this case, for boards which incur a manual set-up, i.e., $k \in T$, a decision is made as to whether or not to process some of the components on the machine).

Computational results are shown in Table 2.1. For problems of size 10×15 (10 boards and 15 components), the Sting heuristic was slightly better than the Greedy heuristic, with an average error of about 3.1% as compared to 4.2% for the Greedy heuristic. For problems of size 10×20, the Greedy heuristics performed better, with an average error of 4.2% versus 4.8% for the Stingy heuristic.

Table 2.1 The case of single machine: computational results for randomly generate problem

No. of Problems Tested	No. of Boards	No. of Components	No. of Slots	Stingy Avg. Error (%)	Stingy Max Error (%)	Greedy Avg. Error (%)	Greedy Max Error (%)	Compound Avg. Error (%)	Compound Max Error (%)
100	10	15	9	3.14	21.53	4.23	31.13	1.74	21.53
30	10	20	12	4.83	15.22	4.19	17.22	3.03	10.27

*Percent deviation from the optimal solution.

In the second case, we develop two different multiple machine heuristics. These heuristics differ in the extent to which we allow part assignment to more than one machine. We assume that the machines are identical, so that $s_i = s_{i+1}$, $c_i = c_{i+1}$, $N_i = N_{i+1}$, $i = 1, \ldots, I\text{-}1$.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Algorithm 2.3: Multiple Machine Stingy Component Algorithm: Version 1

1. Initialization: Let i = 1. Let SK = {1, ..., K}, SJ = {1, ..., J}.
2. "Stingy" Component Assignment: Apply the Stingy Component algorithm to machine i, considering the board set SK and component set SJ.
2. Updating: Remove from SK those boards that are completely processed on machine i, and remove from set SJ those components assigned to machine i, Let i = i + 1, If i < I, return to Step 2.
3. Post-Processing: Given the assignment of components to machines: for each board not completely processed on a single machine, determine the least cost way to produce the board (i.e., on a combination of machines versus machine and manual versus manual only).

   Stop.

The second version of our Multiple Machine Stingy Component algorithm differs only in that, after applying the Single Machine Stingy Component algorithm, we remove from consideration only those components associated with boards that can be completely processed on machine i.

Algorithm 2.4: Multiple Machine Stingy Component Algorithm: Version 2

1. Initialization: Let i = 1. Let SK = {1, ..., K}, SJ = {1, ..., J}.
2. "Stingy" Component Assignment: Apply the Stingy Component algorithm to machine i, considering the board set SK and component set SJ.
2. Updating: Remove from SK those boards that are completely processed on machine i, and remove from set SJ those components that are associated only with boards that can be completely processed on machine i, Let i = i + 1, If i < I, return to Step 2.
3. Post-Processing: Given the assignment of components to machines: for each board not completely processed on a single machine, determine the least cost way to

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

produce the board (i.e., on a combination of machines versus machine and manual versus manual only).

Stop.

Algorithm 2.5: Multiple Machine Greedy Board Algorithm: Version 1

1. Initialization: Let i = 1. Let SK = {1, ..., K}, SJ = {1, ..., J}.
2. "Greedy" Board Assignment: Apply the Greedy Board algorithm to machine i, considering the board set SK and component set SJ.
2. Updating: Remove from SK those boards that are completely processed on machine i, and remove from set SJ those components assigned to machine i, Let i = i + 1, If i < I, return to Step 2.
3. Post-Processing: Given the assignment of components to machines: for each board not completely processed on a single machine, determine the least cost way to produce the board (i.e., on a combination of machines versus machine and manual versus manual only).

   Stop.

Algorithm 2.6: Multiple Machine Greedy Board Algorithm: Version 2

1. Initialization: Let i = 1. Let SK = {1, ..., K}, SJ = {1, ..., J}.
2. "Greedy" Board Assignment: Apply the Greedy Board algorithm to machine i, considering the board set SK and component set SJ.
2. Updating: Remove from SK those boards that are completely processed on machine i, and remove from set SJ those components that are associated only with boards that can be completely processed on machine i, Let i = i + 1, If i < I, return to Step 2.
3. Post-Processing: Given the assignment of components to machines: for each board not completely processed on a single machine, determine the least cost way to

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

produce the board (i.e., on a combination of machines versus machine and manual versus manual only).

Stop.

Results are shown in Table 2.2. For all 30 problems, 20 boards and 100 components, with 35 component slots available on each of the two machines. The Greedy Board algorithms were superior to the Stingy Component algorithms, with Version 2 of the Greedy algorithm yielding the best solutions. The average deviation from J_{min} for the Stingy Component solutions was 7-10%, while the average deviation from J_{min} for the Greedy Board algorithm was 3-5%. These percentage differences are encouragingly small, considering the fact that the measured deviation is based on comparison to an infeasible lower bound.

Table 2.2 The case of two machines: computational results for randomly generated problems

30 Problems Tested		20 Boards		100 Components		35 Slot per Machine	
Stingy Version 1 Avg. Error (%)	Stingy Version 1 Max. Error (%)	Stingy Version 2 Avg. Error (%)	Stingy Version 2 Max. Error (%)	Greedy Version 1 Avg. Error (%)	Greedy Version 1 Max. Error (%)	Greedy Version 2 Avg. Error (%)	Greedy Version 2 Max. Error (%)
9.87	22.48	7.81	20.32	4.65	10.91	3.81	12.31

* Percent deviation from $J_{min} = \sum_k [d_k s_1 + \sum_j v_{jk} c_1]$

In 1998 M. S. Hillier and M. L. Brandeau [9], they solved operation assignment problem is using to the linear programming. The research was inspired by an application at Hewlett-Packard.

BIP1 has (IJK + IK + IJ) variables and (2IJK + IK + I – 1) constraints. For a realistically sized problem of 100 boards, 1,000 components, and just two machines

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

(I = 3), BIP1 will have 303,300 variables and 700,002 constraints. Since all the variables are binary, it is clearly impractical to solve this problem directly.

Fortunately, BIP1 can be reduced considerably. First, we can eliminate all variables x_{ijk} for which $r_{jk} = 0$. These variables will necessarily be zero in any feasible solution by (2.1.1). Furthermore, we can remove the corresponding constraints in (2.1.2) and (2.1.3). Second, we can eliminate (2.1.1) by making the substitution $x_{Ijk} = 1 - \sum_{i=1}^{I-1} x_{ijk}$ for all j and k such that $r_{jk} = 1$. This substitution enforces (2.1.1) while eliminating the variables x_{Ijk}. When there are multiple machines, the constraint $\sum_{i=1}^{I-1} x_{ijk} \leq 1$ must be added in order to ensure that $x_{Ijk} \geq 0$. Third, we can eliminate the variables z_{Ij} by setting them all equal to 1. This is always feasible since there is no capacity constraint on process I and this simplification will not affect optimality.

The result of these reduction is shown in BIP2.

BIP2: Reduced Formulation

Minimize $$\sum_{i=1}^{I-1}\sum_{j}\sum_{k}(c_{ij} - c_{Ij})v_{jk}x_{ijk} + \sum_{i}\sum_{k}s_{ik}d_k y_{ik} + \sum_{j}\sum_{k}c_{Ij}v_{jk} \qquad (2.2)$$

Subject to

$$\sum_{i=1}^{I-1} x_{ijk} \leq 1 \qquad \forall j, k \ni r_{jk} = 1 \qquad (2.2.1)$$

$$y_{ik} \geq x_{ijk} \qquad i = 1, 2, \ldots, I-1;\ \forall j, k \ni r_{jk} = 1 \qquad (2.2.2)$$

$$y_{Ik} \geq 1 - \sum_{i=1}^{I-1} x_{ijk} \qquad \forall j, k \ni r_{jk} = 1 \qquad (2.2.3)$$

$$z_{ij} \geq x_{ijk} \qquad i = 1, 2, \ldots, I-1;\ \forall j, k \ni r_{jk} = 1 \qquad (2.2.4)$$

$$\sum_{j} z_{ij} \leq N_i \qquad i = 1, 2, \ldots, I-1 \qquad (2.2.5)$$

$$x_{ijk} \in \{0, 1\} \qquad i = 1, 2, \ldots, I-1;\ \forall j, k \ni r_{jk} = 1 \qquad (2.2.6)$$

$$y_{ik} \in \{0, 1\} \qquad \forall i, k \qquad (2.2.7)$$

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

$$z_{ij} \in \{0, 1\} \qquad i = 1, 2, \ldots, I-1; \forall j \qquad (2.2.8)$$

(Then and $z_{Ij} = 1$.)

The number of variables and constraints is now a function of the number of component-board pairs such that $r_{jk} = 1$. If the level of component commonality is small, then this number will not be much larger than J. Let the ratio of component-board pairs to components be γ (equivalently, this is the average number of boards that a component is included in). At HP, γ is around 1.25, then BIP2 has $(\gamma IJ + IK + (I - 1)J)$ variables and $(2\gamma IJ + I - 1)$ constraints. For the problem of 100 boards, 1,000 components, and two machine, with γ=1.25, this results in 6,050 variables and 7,502 constraints. This is a substantial improvement over 303,300 variables and 700,002 constraints.

We have not found an efficient algorithm for solving BIP2 to optimality, Therefore, we consider two restricted versions of the problem. The special case of only one machine is formulated in BIP3. Note that the i subscript has been dropped on x_{ijk}, z_{ij} , and N_i since it is always 1.

BIP3: One Automated Process Formulation

Minimize (2.3)

Subject to

$$y_{1k} \geq x_{ijk} \qquad \forall j, k \ni r_{jk} = 1 \qquad (2.3.1)$$

$$y_{2k} \geq 1 - x_{jk} \qquad \forall j, k \ni r_{jk} = 1 \qquad (2.3.2)$$

$$z_j \geq x_{jk} \qquad \forall j, k \ni r_{jk} = 1 \qquad (2.3.3)$$

(2.3.4)

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Table 2.3 Size of the BIPs

Which BIP	Conditions When Appropriate		Size of Problem		Example (I, J, K, γ) = (3, 1000, 100, 1.25)	
	# Machines	Split OK	# Variables	# Constraints	# Variables	# Constraints
BIP1	many	yes	IJK+IK+IJ	2IJK+JK+I-1	303,300	700,002
BIP2	many	yes	γIJ+IK+(I-1)J	2γIJ+I-1	6,050	7,502
BIP3	one	yes	(2γ+1)J+ 2K	3γJ+1	3,700	3,751
BIP4	many	no	(I-1)(K+J)	γIJ+I-1	2,200	3,752
BIP5	one	no	K+J	γJ+I	1,100	1,126

Solution of the single-machine problem

It is easily shown that the component/board allocation problem formulate in the previous section NP-complete in the strong sense (even for the special case where boards cannot be split), since a special case of this problem is equivalent to a three-partition problem. Because the problem is too large to solve using integer programming codes and is NP-complete, Brandeau and Billington [7] developed heuristics that would find a "good" solution (typically within 5 to 10% of optimal). However, there is still hope for solving this problem to optimality. This is because the problem has a special structure. We will show for the single-machine problem that, if the capacity constraints in BIP3 and BIP5 are removed (replaced with a penalty function), the resulting BIP can be solved using linear programming. We exploit this property in a branch-and-bound solution algorithm, the algorithm uses a linear relaxation to obtain feasible solutions, and hence, upper bounds. We now develop the algorithm.

By adding slack and surplus variables and appropriately choosing the coefficients of the vector b and c and matrix A, all the BIPs can be written in the following form:

(P1) Minimize $c^T w$

Subject to $Aw = b$

and $w \in \{0, 1\}$

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

where w = (x y z) is the vector of decision variables. The structure of the matrices A for the single-machine cases, BIP3 and BIP5, are shown in Examples 2.1 and 2.2 for two small problems, where X, Y, Z, and S are the submatrices of A. Notice that all the coefficients are 0, 1, or -1 (all the 0 coefficients have been left blank in the figure for the clarity). Moreover the right-hand sides (not shown) are either 0 or 1.

Example 2.1 Consider the problem has 2 boards and 3 components.

Components on board #1: 1, 2

Components on board #2: 1, 3

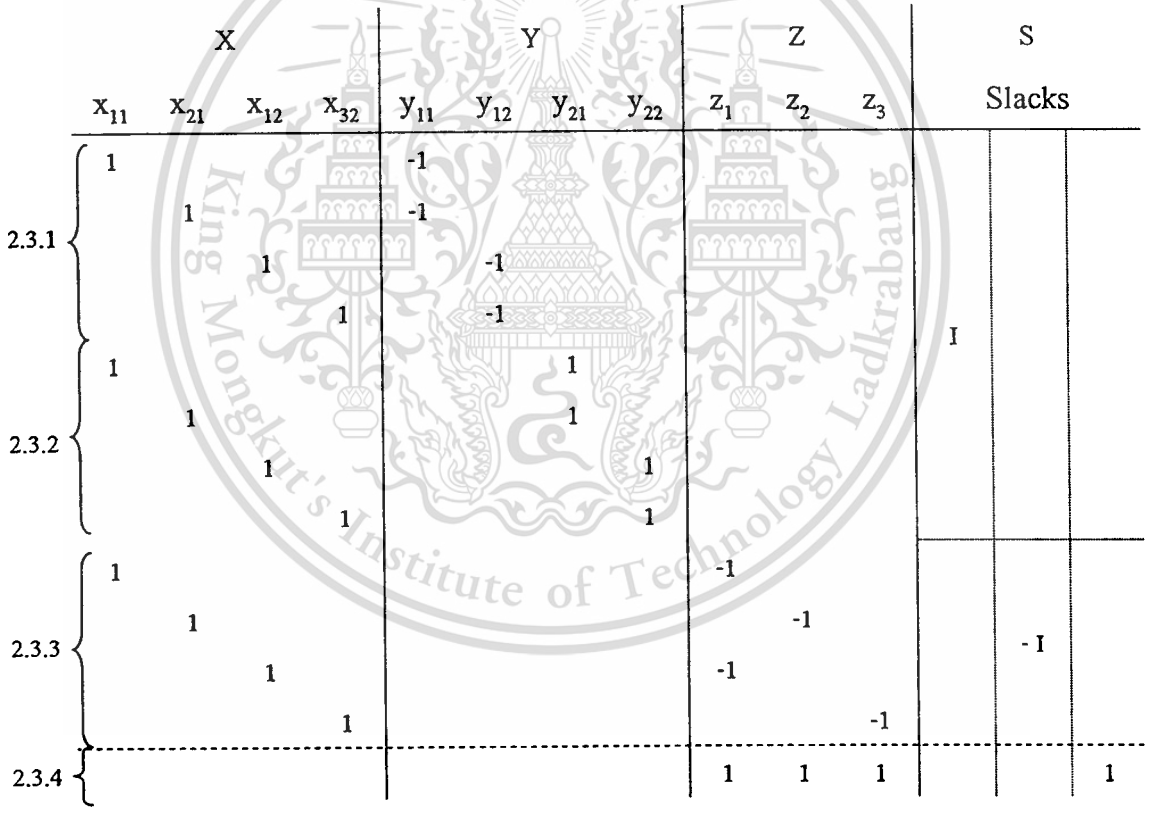

Figure 2.1 Format of the coefficient matrix for BIP3

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.
41542

Example 2.2 Consider the problem has 3 boards and 9 components.

Components on board #1: 1, 2, 3, 4, 5, 6

Components on board #2: 2, 6, 7, 8

Components on board #3: 4, 7, 9

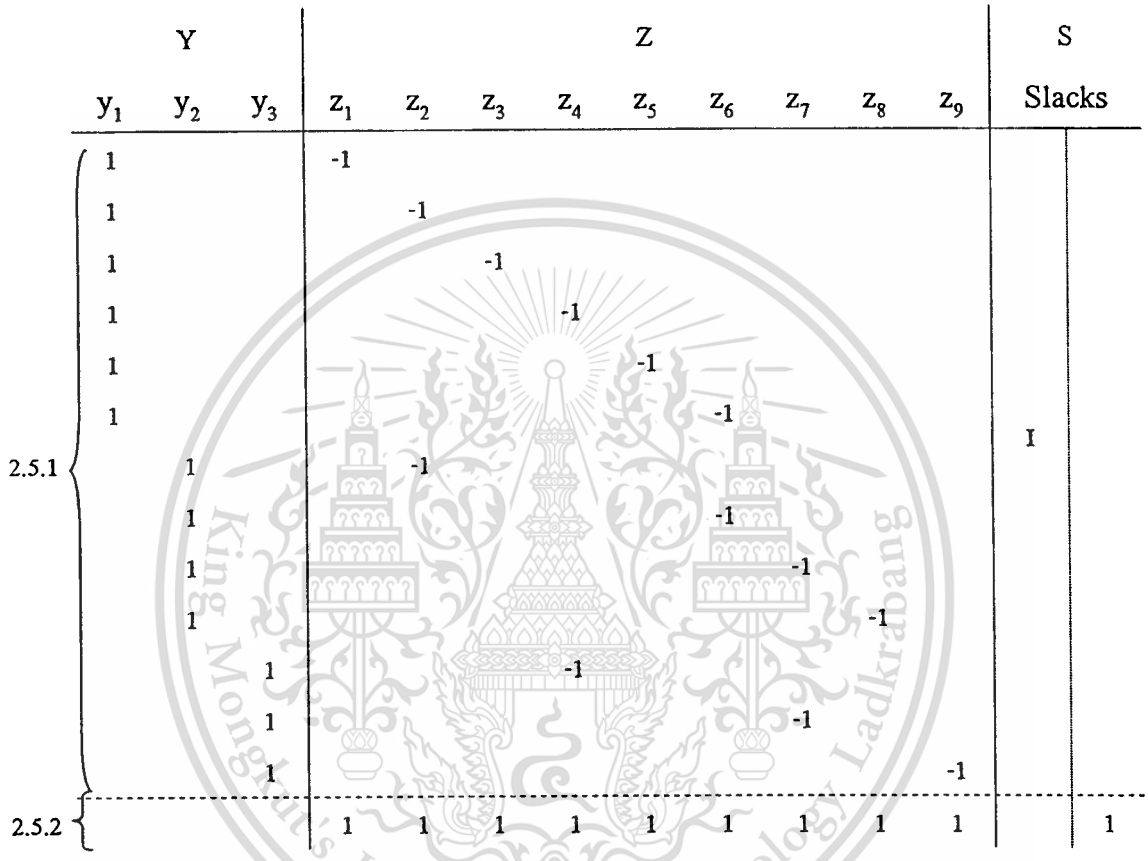

Figure 2.2 Format of the coefficient matrix for BIP5

If we could show that the matrix A is totally unimodular, we could drop the integrality constraints and solve the problem as an LP, The result would be an integral optimal solution. Since LP codes can solve vastly larger problems than can MIP or BIP codes, even the large problems formulated in the previous section could be readily solved.

Unfortunately, the matrix A is not totally unimodular for any of the cases. However, if we remove just the capacity constraint from BIP3 or BIP5, the resulting

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

coefficient matrices are totally unimodular. We can exploit this property in a branch-and-bound solution algorithm.

Let M be the coefficient matrices for the constraints in BIP3 and BIP5 with the capacity constraint removed, and let the vector inequality $g(w) \leq N$ represent the capacity constraints. Then BIP3 and BIP5 can be written as follows:

(P2) Minimize $c^T w$

Subject to $Mw = b$

$g(w) \leq N$

and $w \in \{0, 1\}$

In Examples 2.1 and 2.2, M consists of all of the coefficients above the dashed line, while the capacity constraint coefficients are below the dotted line.

Properties of the matrices M:

(a) All elements are +1, 0, or –1.

(b) Columns in S have only one nonzero element.

(c) All nonzero elements in a column have the same sign.

(d) If two rows have a nonzero element in the same column of X, then they do not have any nonzero elements in the same column of Y, Z, or S.

(e) In each row, there is for (BIP3): one 1 in X, one nonzero element in Y or Z, and one nonzero element in S. (BIP5): nothing in X, one 1 in Y, one –1 in Z, and one 1 in S.

Theorem 2.1 The matrix M formed by removing the capacity constraint from the coefficient matrix A of BIP3 is totally unimodular.

Theorem 2.2 The matrix M formed by removing the capacity constraint from the coefficient matrix A of BIP5 is totally unimodular.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

By Theorem 2.1 and 2.2, if the machine capacity constraint is removed, the resulting relaxations of BIP3 and BIP5 can be solved using linear programming rather than binary integer programming techniques. However, by removing the capacity constraint we have trivialized the problem. The LP solution will assign all of the boards and components to the fastest (i.e., cheapest) machine, most likely exceeding the capacity of the machine in the original problem.

Therefore, we add a penalty to the objective function for violating the capacity constraint, applying Lagrangian relaxation with a fixed $\lambda \geq 0$ gives the following:

(P3) Minimize $c^T w + \lambda(g(w) - N)$

Subject to $Mw = b$

and $w \in \{0, 1\}$

Since M is totally unimodular, we can drop the binary constraint and solve the following LP:

(P4) Minimize $c^T w + \lambda(g(w) - N)$

Subject to $Mw = b$

and $0 \leq w \leq 1$

The optimal solution for P4 will also be an optimal solution for P3. However, we seek an optimal solution for P2. We know from the theory of Lagrangian relaxation that if $\lambda(g(w) - N) = 0$ for an optimal solution for P4, then this solution is also optimal for P2; and if $g(w) \leq N$ for an optimal solution for P4, then this solution is also feasible for P2.

If we can choose λ such that the machine capacity is exactly met, the optimal solution to P4 will also be optimal for P2. Although it is quite possible that there will exist a λ such that there exists an optimal solution to P4 with $g(w) - N = 0$, it is not

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

guaranteed because of the potential duality gap. Consider the following relaxation of P2:

(P5) Minimize $c^T w$

Subject to $Mw = b$

$g(w) \leq N$

and $0 \leq w \leq 1$

P5 is identical to P2, except that the integrality constraint has been relaxed. If P5 is solved to optimality, shadow prices will be generated for the capacity constraint. A shadow price represents the value of increasing the machine capacity by one. This is exactly the value that we should choose for λ. This would charge just enough for each component so as to meet the capacity constraint (approximately), but go no further.

A good way to choose the value of λ for P4 is therefore to first solve P5 and set λ equal to the value of the shadow price for the capacity constraint. It might seem inefficient to have to solve two LPs (P4 and P5) in order to determine a bound on P2. However, note that any feasible solution for P5 is also a feasible solution for P4, and that the optimal solution for P5 is likely to be very close (if not identical) to the optimal solution for P4. Therefore, in order to solve P4, we can start at the optimal solution for P5 and drastically reduce the number of iterations required. In fact, if the optimal solution for P5 in integral, there is no need to solve P4 at all as the solution is also optimal for P2.

If the given λ for P4 does not yield $g(w) - N = 0$, then sensitivity analysis techniques can be used to adjust the solution for slight variations in λ. By choosing the appropriate value of λ, one can find a feasible solution for P2 that either exactly meets the capacity constraint ($g(w) = N$) or else comes very close ($g(w) < N$, but not by much). In the former case, we are done since we have found an optimal solution for P2. In the latter case, we have hopefully found a good solution for P2. Since the

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

solution is feasible, it giver an upper bound on the cost of the optimal solution for P2 (calculated using the objective function value of the solution to P4, with the term $\lambda(g(w) - N))$ removed).

In the case where $g(w) < N$, an even better solution (and hence a tighter upper bound) can usually be found by then adding additional boards and/or components to the machine in a greedy fashion until the capacity is exactly met. Algorithm 2.7 presents such an algorithm. The algorithm adds boards (along with all of the associated components) to the machine in descending order of the cost saving per component added. In the case where boards are allowed to be split across processes, the algorithm continues by adding components to the machine in descending order of cost savings. The computational time for this algorithm is trivial relative to that of solving the linear programs.

By solving P5, one can obtain a lower bound on the solution to P2. Then, by solving P4 (and then applying the algorithm in Algorithm 2.7), a feasible solution, and hence an upper bound, can be obtained. To find an optimal solution for P2, one can apply branch-and-bound, branching on the nonintegral variables from the optimal solution to P5, and bounding using the solutions of P5 and P4. This algorithm is formally specified in Algorithm 2.8.

An effective heuristic is to skip the branch-and-bound step. In this case, we find the optimal solution for P4 and then apply the algorithm in Algorithm 2.7 to improve this solution (Steps 1 to 3 in Algorithm 2.8). This heuristic will be called the Lagrangian relaxation heuristic.

The solution techniques presented above were applied to several data sets. The first set is a small problem (30 boards and 316 components), the other three data sets are for much larger problems, that were produced randomly in such a way as to maintain the same characteristics. The capacity of the machine was 220 and 700 for the 30 board and 100 board problems, respectively. For all the data sets, the component commonality factor γ was approximately 1.25. the setup and insertion

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

costs used in the model were the average board setup time (in seconds) and the average component insertion time (in seconds). A comparison of Brandeau and Billington's "Greedy Board" heuristic, the Lagrangian relaxation heuristic, and the branch-and-bound optimal solution technique (Algorithm 2.8) are shown below. Table 2.4 considers the case where boards are allowed to be split (BIP3), and Table 2.5 considers the case where boards are not allowed to be split (BIP5).

Algorithm 2.7: Improving the upper bound

Step 1 Let $a = N - \sum_{j=1}^{J} z_j$ be the number of empty slots remaining on the machine.

Step 2 For each board, k, let $\alpha_k = \sum_{j=1}^{J} [r_{jk} - z_j]^+$ be the number of components that would need to be added to the machine in order to produce board k entirely on the machine. Let $\Omega = \{k \mid 1 \le \alpha_k \le a\}$ be the set of al boards that can be moved to the machine. If $\Omega = \varnothing$, then go to Step 5.

Step 3 For each board, $k \in \Omega$, let

$$\gamma_k = \begin{cases} s_{2k} d_k \gamma_{2k} - s_{1k} d_k (1 - y_{1k}) + \sum_{j=1}^{J} [(c_{2j} - c_{1j}) v_{jk} (1 - x_{jk})] & \text{for BIP3} \\ \left[(s_{2k} - s_{1k}) d_k + \sum_{j=1}^{J} (c_{2j} - c_{1j}) v_{jk} \right] (1 - y_k) & \text{for BIP5} \end{cases}$$

be the cost saving achieved by producing board k entirely on the machine.

Step 4 Let $k^* = \arg\max_{k \in \Omega} (\gamma_k / \alpha_k)$ be that board which maximizes the cost savings per component slot used when added to the machine. All board k^* and all of its components to the machine (break ties arbitrarily). Go to Step 1.

Step 5 If a = 0 or boards cannot be split (i.e., BIP5), then stop. Otherwise continue to Step 6.

Step 6 Let $\Psi = \{k \mid y_{1k} = y_{2k} = 1\}$ be the set of all boards that are split across the machine and the manual process. If $\Psi = \varnothing$, then stop. Otherwise continue to Step 7.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Step 7 For each component, j, let $\varphi_j = \sum_{k \in \Psi} (c_{1j} - c_{2j}) v_{jk} (1 - x_{jk})$ be the cost savings achieved by adding component j to the machine. Add to the machine the a components with the a highest values of φ_j (break ties arbitrarily).

Algorithm 2.8: Solving P2 optimally.

Step 1 Solve P5. If the solution is integral, then stop; the solution is also optimal for P2. Otherwise, the cost of the solution gives a lower bound on the cost of the optimal solution for P2.

Step 2 Set λ equal to the shadow prices for the capacity constraint, and solve P4. Begin pivoting at the solution for P5.

Step 3 a) If $g(w) = N$, then stop; the solution is optimal for P2.

b) If $g(w) < N$, then apply the Algorithm 2.7

c) If $g(w) > N$, then use sensitivity analysis to adjust λ until $g(w) \leq N$.

(i) If $g(w) = N$, then stop; the solution is optimal for P2.

(ii) If $g(w) < N$, then apply the Algorithm 2.7

Step 4 Apply branch-and-bound as follows.

a) Branch on variables which were nonintegral in the solution of P5 (give preference to the y_{ik} variables, as they will have the greatest effect) by setting them equal to 0 and 1 in the two respective branches.

b) Bound using Step 1 through 3.

Algorithm 2.9: The multiple-machine Lagrangian relaxation process.

Step 1 Let $\mathcal{K} = 1, \ldots, K$ be the set of all boards. Let $\mathfrak{I} = 1, \ldots, I\text{-}1$ be the set of all machines.

Step 2 Let $i^* = \arg\min_{i \in \mathfrak{I}} \left(\sum_{k \in \mathcal{K}} (s_{ik} + \sum_j n_{jk} c_{ij}) \right)$ be the "fastest" machine (defined as that machine which could produce the entire set of boards the fastest), with ties broken arbitrarily.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Step 3 Apply the Algorithm 2.8 for the single-machine problem, using machine i^* as the single machine, and considering all boards in the set K.

Step 4 Remove from the set K all boards assigned to machine i^*. Let $J = J - i^*$.

Step 5 If $J \neq \varnothing$ then go to Step 3. Otherwise, stop, and assign all boards in K to the manual process.

Table 2.4 Comparison of the Different Solution Techniques (when boards may be split)

Data Set	Number of Boards	Number of Components	Brandeau & Billington "Greedy Board"			Lagrangian Relaxation Heuristic			Optimal Solution	
			Cost	Above Optimal (%)	CPU Time (Sec.)	Cost	Above Optimal (%)	CPU Time (Sec.)	Cost	CPU Time (Sec.)
1	30	316	765.5	2.50	0.4	747.3	0.060	22.2	746.9	32.2
2	100	1,056	2,949.6	2.91	11.4	2,866.2	0.0	201.3	2,866.2	334.9
3	100	972	3,297.9	1.75	10.6	3,241.2	0.0022	224.4	3,241.1	416.4
4	100	1,084	3,160.5	3.53	11.4	3,052.7	0.0	228.7	3,052.7	344.3

Table 2.5 Comparison of the Different Solution Techniques (when boards may not be split)

Data Set	Number of Boards	Number of Components	Brandeau & Billington "Greedy Board"			Lagrangian Relaxation Heuristic			Optimal Solution	
			Cost	Above Optimal (%)	CPU Time (Sec.)	Cost	Above Optimal (%)	CPU Time (Sec.)	Cost	CPU Time (Sec.)
1	30	316	776.2	2.10	0.4	760.3	0.0	5.0	760.3	5.0
2	100	1,056	2,979.9	1.47	11.2	2,936.6	0.0	34.8	2,936.6	48.3
3	100	972	3,355.9	1.16	10.5	3,317.3	0.0018	33.8	3,317.3	130.3
4	100	1,084	3,200.3	1.84	11.3	3,142.4	0.0	37.5	3,142.4	197.5

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Table 2.6 Robustness Result (boards may be split)

Data Set	Number of Boards	Number of Components	Percent Above Lagrangian Relaxation with Perfect Information			
			Cost of Greedy Board		Cost of Lagrangian Relaxation Heuristic	
			Maximum	Average	Maximum	Average
1	30	316	6.26	5.56	2.27	1.50
2	100	1,056	4.57	5.50	1.31	0.71
3	100	972	4.64	4.02	1.10	0.63
4	100	1,084	6.50	5.78	1.54	0.93

Table 2.7 Robustness Result (boards may not be split)

Data Set	Number of Boards	Number of Components	Percent Above Lagrangian Relaxation with Perfect Information			
			Cost of Greedy Board		Cost of Lagrangian Relaxation Heuristic	
			Maximum	Average	Maximum	Average
1	30	316	4.73	3.71	2.43	1.37
2	100	1,056	3.30	2.60	1.71	1.20
3	100	972	2.38	1.89	1.34	0.72
4	100	1,084	3.64	2.74	1.48	0.90

The cost of Brandeau and Billington's "Greedy Board" heuristic was between 1% and 4% above optimal. On the other hand, the Lagrangian relaxation heuristic usually yielded an optimal solution, and if not, was exceptionally close (less than 0.003% for all of the large data sets). The difference between the costs of the Lagrangian relaxation heuristic and the optimal solution is so small that the branch-and-bound technique does not seem necessary for most practical purposes. Table 2.6 considers the case where boards are allowed to be split (BIP3), and Table 2.7 considers the case where boards are not allowed to be split (BIP5). Clearly, the Lagrangian relaxation heuristic provides a very robust solution. On average, the cost of not having perfect information is only about 1%.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Solution of the multiple-machine problem

Unfortunately, the technique used to solve the single-machine problem to optimality cannot be applied directly to the multiple-machine problem. The reason is that even when the capacity constraints are removed, neither BIP2 nor BIP4 is totally unimodular. However, for the case when boards are not allowed to be split across more than one process, the techniques from the preceding section can be extended to find either an optimal or a good heuristic solution for the multiple-machine problem.

A feasible solution, and hence an upper bound on the optimal cost, can be found by extending the Lagrangian relaxation heuristic to the case of multiple machines. Consider a problem with $I > 2$ (i.e., more than one machine). Starting with the fastest machine (defined as the machine that could produce the entire set of boards the fastest), we solve the one-machine problem where boards cannot be split across processes. The boards assigned to the first machine are then removed from further consideration. Then, the single-machine problem is solved again with the second fastest machine. This process is repeated until all of the machines have been considered. Then all unassigned boards are assigned to the manual process. A formal description of the multiple-machine Lagrangian relaxation heuristic is given Algorithm 2.9

The Lagrangian relaxation heuristic gives an upper bound on the optimal cost. A lower bound can be found by solving the linear relaxation of BIP4. Then, branch-and-bound can be performed in order to find an optimal solution to the multiple-machine problem by branching on the nonintegral values in the solution to BIP4.

Both the Lagrangian relaxation heuristic and the optimal solution technique were applied to a two-machine problem with the same data sets used for the single-machine problem (except that component capacities were halved in order to avoid a trivial solution). One machine was faster than the other, but both were faster than the manual process. Table 2.8 considers the cost of Brandeau and Billington's Greedy Board heuristic was between 0% and 8% above the optimal solution. The Lagrangian

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

relaxation heuristic gave solutions within 0.3% of the optimal solution. The CPU time for the largest problems was between two and three minutes for the Lagrangian relaxation heuristic, and between 25 and 40 minutes for the optimal solution technique Even the computation time for the optimal solution technique is not too long when one considers the planning horizon of the problem.

As for the single-machine case, we next tested the robustness of the solution obtained. We again generated 10 random instances of demand data, and compared the cost of the original solutions (which used the expected demand data) with the Lagrangian relaxation heuristic solution given perfect information (i.e., re-solved using the actual demand data). The average and worst-case results (across the 10 different demand instances) are shown in Table 2.9 for both the Greedy Board heuristic and the Lagrangian relaxation heuristic. As in the single-machine case, the Lagrangian relaxation heuristic provides a very robust solution. On average, the cost of not having perfect information is about 1.5%.

Table 2.8 Comparison of the Different Solution Techniques for Two-machine Problems

Data Set	Number of Boards	Number of Components	Brandeau & Billington "Greedy Board"			Lagrangian Relaxation Heuristic			Optimal Solution	
			Cost	Above Optimal (%)	CPU Time (Sec.)	Cost	Above Optimal (%)	CPU Time (Sec.)	Cost	CPU Time (Sec.)
1	10	95	114.1	7.46	< 0.1	106.2	0.00	3.1	106.2	3.6
2	10	114	279.5	0.00	< 0.1	279.5	0.00	3.9	279.5	4.8
3	30	316	677.6	2.67	0.4	662.0	0.29	17.7	660.0	50.7
4	100	1,056	2,483.3	3.07	10.2	2,413.5	0.19	162.0	2,409.0	1,659.7
5	100	972	2,776.2	1.99	9.0	2,727.1	0.18	167.7	2,722.1	2,403.1
6	100	1,084	2,768.0	3.29	9.9	2,684.3	0.17	161.5	2,679.8	1,561.4

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Table 2.9 Robustness Result for Two-machine Problems

Data Set	Number of Boards	Number of Components	Percent Above Lagrangian Relaxation with Perfect Information			
			Cost of Greedy Board		Cost of Lagrangian Relaxation Heuristic	
			Maximum (%)	Average (%)	Maximum (%)	Average (%)
1	30	316	6.28	3.47	4.28	1.18
2	100	1,056	6.38	4.26	2.51	1.57
3	100	972	4.42	3.03	2.10	1.02
4	100	1,084	6.01	5.05	2.96	1.92

King Mongkut's Institute of Technology Ladkrabang

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 3

A VARIABLE DECOMPOSITION ALGORITHM FOR SOLVING THE OPERATION ASSIGNMENT PROBLEM

In this chapter , we state the algorithm for solving the operation assignment problem by Variable Decomposition method.

3.1 VARIABLE DECOMPOSITION

From the original problem BIP1, we can modify as follows:

Minimize $$\sum_i \sum_j \sum_k c_{ij} v_{jk} x_{ijk} + \sum_i \sum_k s_{ik} d_k y_{ik} \quad (3.1)$$

Subject to

$$\sum_i x_{ijk} - r_{jk} = 0 \qquad \forall j, k \quad (3.1.1)$$

$$x_{ijk} - y_{ik} \leq 0 \qquad \forall i, j, k \quad (3.1.2)$$

$$x_{ijk} - z_{ij} \leq 0 \qquad \forall i, j, k \quad (3.1.3)$$

$$\sum_j z_{ij} - N_i \leq 0 \qquad i = 1, 2, \ldots, I-1 \quad (3.1.4)$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j, k \quad (3.1.5)$$

$$y_{ik} \in \{0, 1\} \qquad \forall i, k \quad (3.1.6)$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j \quad (3.1.7)$$

Generically, the model can be written as:

Minimize $f(x, y, z)$

Subject to $g(x, y, z) \leq 0$

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

$$x \in \{0, 1\}$$

$$y \in \{0, 1\}$$

$$z \in \{0, 1\}$$

where $f(x, y, z)$ replaces (3.1) and $g(x, y, z) \leq 0$ replaces (3.1.1) – (3.1.4).

.

This research use a variable decomposition method to solve the problem. The key idea of decomposition is that it breaks a difficult problem into smaller and easier sub-problems. The sub-problems are then solved *iteratively* and their solutions coordinated in a proper way to drive the overall solution to optimality of the original problem. How the problem is decomposed, and how the sub-problems are solved and their solutions coordinated depend on the structure of the problems themselves. Decomposition has been the main subject of research in dealing large scale optimization problems for the last few decades. The text by Lasdon [6] and the book by Singh and Titli [8] have captured the essence of the result in decomposition that had been obtained up to that point in time. In Problem (3.1), there is also a special structure that can be taken advantage of so that a customized form of decomposition can be applied. First, if we attempt to completely decompose the three sets of variables, x, y and z, we would get a three-level decomposition scheme as shown in Diagram 3.1.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

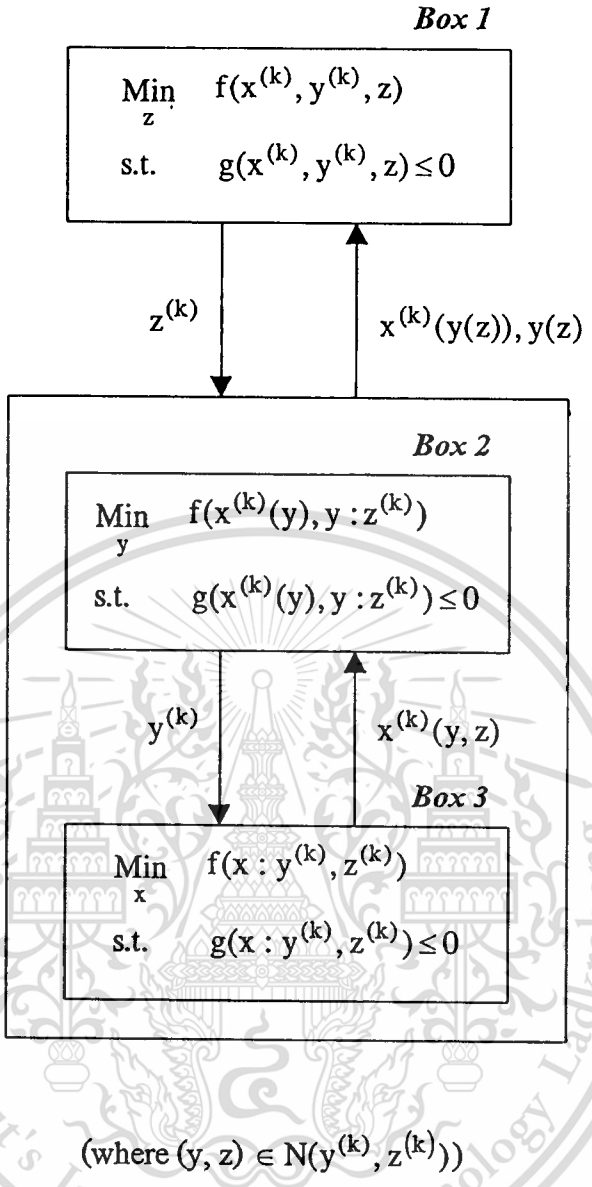

Diagram 3.1 A Completely Decomposed form of Problem (3.1)

To solve Problem (3.1) using the decomposition scheme above, we need to do the following:

1. Initialize the values of $z^{(0)}$ and $y^{(0)}$, and solve Box 3 for $x^{(0)}(y, z)$. Set k=0.
2. Set $z = z^{(k)}$ and $y = y^{(0)}$ in function f(x, y, z).
3. Use Box 3 to find the optimal $x^{(k)}$ as a function of y and z, $x^{(k)}(y, z)$ in the neighborhood of $(y^{(k)}, z^{(k)})$.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

4. Use Box 2 to find the optimal $y^{(k+1)}$ (with x replaced by $x^{(k)}(y, z)$) as a function of z, $y^{(k+1)}(z)$ in the neighborhood of $z^{(k)}$. Then return to Box 3 and iterate between Box 2 and 3 until an optimal $y^{(k+1)}(z)$ and $x^{(k+1)}(y^{(k+1)}(z), z)$. Then go up to Box 1 with these two functions of z.

5. Solve the revised Box 1 with x and y replaced by $x^{(k+1)}(y^{(k+1)}(z), z)$ and $y^{(k+1)}(z)$ to find a new level of $z^{(k+1)}$.

6. If $(x^{(k+1)}(y^{(k+1)}(z^{(k+1)}), z^{(k+1)}), y^{(k+1)}(z^{(k+1)}),z^{(k+1)})$, the repeat Step 3.

However, the above scheme would not be effective and efficient, since it is generally difficult to solve Box 2 to obtain a functional form $y^{(k+1)}(z)$, and the iterative procedure would most likely take too many iterations and it might stop at a point that is not even local optimal $y^{(k+1)}(z)$.

To get a more effective decomposition scheme, we note that (3.1.2) and (3.1.3) represents the coupling of individual x-variables with individual y-variables and z-variables respectively. Also (3.1.1) represents the intra-coupling of x-variables with respect to the index i. Likewise, (3.1.4) represents the intra-coupling of z-variables with respect to the index j. We observe that upon fixing y and z, the resulting problem is completely decomposable into JK independent sub-problems, each one of which can be solved by inspection yielding individual x_{ijk} as a function of y and z. This helps eliminate the IJK variables of x_{ijk} from the picture, and all we need to take care of is the variables y and z, after each x_{ijk} is replaced by $x_{ijk}(y, z)$. This suggests a two-level variable decomposition scheme as shown in Diagram 3.2 below:

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

$$x_{ijk} \le \min(y_{ik}, z_{ij}) \quad \forall i, j, k \tag{3.3.2}$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j, k \tag{3.3.3}$$

On fixing each of these subproblems can then be solved easily by inspection as the following lemma will indicate:

Lemma 3.1 For each (j, k), the optimal solution of Problem (3.3) is as follows:

Case 1 when $r_{jk} = 0$: $x_{ijk} = 0 \ \forall i$

Case 2 when $r_{jk} = 1$: $x_{i'jk} = 1$ and $x_{ijk} = 0, \ \forall i \neq i'$

where i' is the index of the process $c_{ij}v_{jk}$ is minimum among $i \in I_{jk}$ and

$I_{jk} = \{i \mid \min(y_{ik}, z_{ij}) = 1\}$

Proof:

Case 1 when $r_{jk} = 0$: It is clear from (3.3.1) and (3.3.3) that when $r_{jk} = 0$, each $x_{ijk} = 0$ will necessary have to be equal to 0 to satisfy the two constraints. Clearly (3.3.2) is also satisfied by this single feasible solution. It is therefore optimal.

Case 2 when $r_{jk} = 1$: Again (3.3.1) and (3.3.3) dictate that one and only one x_{ijk} is equal to 1, and the remaining ones must be equal to 0. Symbolically, let $x_{ijk} = 1$, when $i \neq i'$ and $x_{ijk} = 0 \ \forall i \neq i'$. The question is which of the i's is i' (so that $x_{i'jk}$ is equal to 1). Clearly (3.3.2) requires that i' must come from those i such that $\min(y_{ik}, z_{ij}) = 1$ (i.e. both y_{ik} and z_{ij} equal 1), for otherwise x_{ijk} must be 0 if $\min(y_{ik}, z_{ij}) = 0$. Let I_{jk} be the collection of all i's such that $\min(y_{ik}, z_{ij}) = 1$. That is i' must be selected from I_{jk}. Now to minimize the objective function in (3.3), it is clear that we should select i' (to set $x_{i'jk} = 1$) such that $c_{ij}v_{jk}$ is minimum among all $i \in I_{jk}$. This completes the proof of Lemma 3.1.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

$$x_{ijk} \le \min(y_{ik}, z_{ij}) \quad \forall i, j, k \tag{3.3.2}$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j, k \tag{3.3.3}$$

On fixing each of these subproblems can then be solved easily by inspection as the following lemma will indicate:

Lemma 3.1 For each (j, k), the optimal solution of Problem (3.3) is as follows:

Case 1 when $r_{jk} = 0$: $x_{ijk} = 0 \ \forall i$

Case 2 when $r_{jk} = 1$: $x_{i'jk} = 1$ and $x_{ijk} = 0, \ \forall i \neq i'$

where i' is the index of the process where $c_{ij}v_{jk}$ is minimum among $i \in I_{jk}$

and $I_{jk} = \{i \mid \min(y_{ik}, z_{ij}) = 1\}$

Proof:

Case 1 when $r_{jk} = 0$: It is clear from (3.3.1) and (3.3.3) that when $r_{jk} = 0$, each $x_{ijk} = 0$ will necessary have to be equal to 0 to satisfy the two constraints. Clearly (3.3.2) is also satisfied by this single feasible solution. It is therefore optimal.

Case 2 when $r_{jk} = 1$: Again (3.3.1) and (3.3.3) dictate that one and only one x_{ijk} is equal to 1, and the remaining ones must be equal to 0. Symbolically, let $x_{ijk} = 1$, when $i \neq i'$ and $x_{ijk} = 0 \ \forall i \neq i'$. The question is which of the i's is i' (so that $x_{i'jk}$ is equal to 1). Clearly (3.3.2) requires that i' must come from those i such that $\min(y_{ik}, z_{ij}) = 1$ (i.e. both y_{ik} and z_{ij} equal 1), for otherwise x_{ijk} must be 0 if $\min(y_{ik}, z_{ij}) = 0$. Let I_{jk} be the collection of all i's such that $\min(y_{ik}, z_{ij}) = 1$. That is i' must be selected from I_{jk}. Now to minimize the objective function in (3.3), it is clear that we should select i' (to set $x_{i'jk} = 1$) such that $c_{ij}v_{jk}$ is minimum among all $i \in I_{jk}$. This completes the proof of Lemma 3.1.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

of z, we consider a special class of Problem (3.4), where (3.4.3) is converted to equality as shown below:

$$\text{Minimize} \quad \sum_i \sum_j \sum_k c_{ij} v_{jk} x_{ijk}(y, z) + \sum_i \sum_k s_{ik} d_k y_{ik} \tag{3.5}$$

Subject to

$$x_{ijk}(y, z) - y_{ik} \leq 0 \qquad \forall i, j, k \tag{3.5.1}$$

$$x_{ijk}(y, z) - z_{ij} \leq 0 \qquad \forall i, j, k \tag{3.5.2}$$

$$\sum_j z_{ij} = N_i \qquad i = 1, 2, \ldots, I-1 \tag{3.5.3}$$

$$y_{ik} \in \{0, 1\} \qquad \forall i, k \tag{3.5.4}$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j \tag{3.5.5}$$

The following lemma shows that a solution of Problem (3.5) will also solve Problem (3.4).

Lemma 3.2 For any fixed y, say $y = y^*$, if (y^*, z^*) solves Problem (3.5), then it must also solves Problem (3.4) when y is fixed at the same value y^*.

Proof:

We again note that (3.4.3) is the only constraint in Problem (3.4) that couples the components (across the index j). Lemma 3.1 also indicates that y and z are independent variables and $x_{ijk}(y_{ik}, z_{ij})$ is equal to 0 for all i except possibly $x_{i'jk}$ $(y_{ik}, z_{ij}) = 1$ when $r_{jk} = 1$ and $c_{i'j}v_{jk} = \min\{c_{ij}v_{jk}\}$ among all $i \in I_{jk} = \{ i \mid \min (y_{ik}, z_{ij}) = 1\}$. For a given $y = y^*$, let (y^*, z^*) solve Problem (3.5), and let (y^*, z') solve Problem (3.4) with (3.4.3) replaced by $\sum_j z_{ij} < N_i$ for some $i = 1, 2, \ldots, I-1$. Let $I^*_{jk} = \{ i \mid \min (y^*_{ik}, z^*_{ij}) = 1\}$ and $I'_{jk} = \{ i \mid \min (y^*_{ik}, z'_{ij}) = 1\}$. Clearly $I'_{jk} \subseteq I^*_{jk}$ since the

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

number of $z_{ij} = 1$ in $\sum_j z_{ij} < N_i$ cannot be greater than the number of $z_{ij} = 1$ in $\sum_j z_{ij} = N_i$. Thus

$$c_{i''j}v_{jk} = \min_{i \in I'_{jk}} \{c_{ij}v_{jk}\} \geq c_{i'j}v_{jk} = \min_{i \in I^*_{jk}} \{c_{ij}v_{jk}\}$$

Hence the optimal objective value of Problem (3.4) with $\sum_j z_{ij} < N_i$ for a given y cannot be better that the optimal objective value of Problem (3.5) for the same y. Thus any solution of Problem (3.5) for a given y must also solve Problem (3.4) for the same value of y. This completes the proof of Lemma 3.2.

As a collorary of Lemma 3.2, it is clear that there always exists an optimal solution to Problem (3.4) such that (3.4.3) is active for each i (i.e. $\sum_j z_{ij} = N_i$). This suggests that we can simply solve Problem (3.5) instead of Problem (3.4) to make is simpler.

Lemma 3.2 indicates that in enumerating y values (i.e. setting y for each possible combination of y for given board k, which is not too large in number as discussed above), we can solve Problem (3.5) instead of Problem (3.4). Hence the number of combinations of z variables will be reduced. In fact, on using Problem (3.5), the number of combinations of z values to consider for each process i that is used (i.e. $y_{ik} = 1$, for some board k), is equal to $\binom{N}{N_i}$, and the number of combinations of z for all processes except the manual process is $\prod_{i=1}^{I-1}\binom{N}{N_i}$. This represents a considerable reduction in the number of combinations that needs to be considered in Problem (3.4), which is equal to $\prod_{i=1}^{I-1}\left(\sum_{n=1}^{N_i}\binom{N}{n}\right)$. In this research, we use the results of Lemma 3.1 and Lemma 3.2, and a complete enumeration to solve Problem (3.5) to generate an optimal solution of Problem 3.1. An algorithm to implement this is outlined below and the testing of the algorithm is done in the next chapter.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3.2 THE ALGORITHM

In this section, we state the variable decomposition algorithm described above

Step 0 Input all constants including r_{jk}'s.

Step 1 Do for each possible value of y

Step 2 Do for each possible value of z

Step 3 For each (y, z) set in steps 1 and 2 above, solve Problem (3.3) by using Lemma 3.1, i.e. set x_{ijk} for each j and k as follows:

$$\text{if } r_{jk} = 0, \text{ then } x_{ijk} = 0, \ \forall i$$

$$\text{and if } r_{jk} = 1, \text{ then } x_{i'jk} = 1 \text{ and } x_{ijk} = 0, \ \forall i \neq i'$$

$$\text{where } i' = \arg\{\min_{i \in I_{jk}} c_{ij} v_{jk}\}$$

$$I_{jk} = \{i \mid \min(y_{jk}, z_{ik}) = 1\}$$

Step 4 Repeat Step 2

Step 5 Repeat Step 1

It is clear from Lemmas 3.1 and 3.2 and a complete enumeration process of solving Problem (3.5) that the algorithm will yield an optimal solution to Problem (3.1).

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 4

SOME EXAMPLES FOR SOLVING THE OPERATION ASSIGNMENT PROBLEM

In this section, we consider some examples for solving operation assignment problems by Variable Decomposition method and application problems.

4.1 EXAMPLE FOR SOLVING THE PROBLEM BY VARIABLE DECOMPOSITION METHOD

Example 4.1 We consider problem of 2 boards, 4 components and 1 machine.

Input data

$i = 2,\ j = 4,\ k = 2$

r_{jk}	k=1	2
j=1	1	1
2	1	0
3	0	1
4	1	1

n_{jk}	k=1	2
j=1	3	4
2	5	0
3	0	2
4	4	2

s_{ik}	k=1	2
i=1	3	2
2	2	1

c_{ij}	j=1	2	3	4	N_i	d_k
i=1	1	1	2	1	3	8
2	2	3	4	2	4	6

Compute $v_{jk} = n_{jk} d_k$

v_{jk}	k=1	2
j=1	24	24
2	40	0
3	0	12
4	32	12

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Each possible value of y

y1	y2	y3
1	0	1
0	1	1

Each possible value of z

Case1

z_{ij}	j=1	2	3	4
i= 1	1	1	1	0
2	1	1	1	1

Case2

z_{ij}	j=1	2	3	4
i= 1	1	1	0	1
2	1	1	1	1

Case 3

z_{ij}	j=1	2	3	4
i= 1	1	0	1	1
2	1	1	1	1

Case 4

z_{ij}	j=1	2	3	4
i= 1	0	1	1	1
2	1	1	1	1

Solve problem

Case 1

k=1 $c_{11}v_{11}z_{11}y1_1$ $s_{11}d_1y1_1$ k=2

	24	40	0	0	24	B	24	0	24	0	12
$c_{21}v_{11}z_{21}y1_2$	0	0	0	0	0		0	0	0	0	0
	A				A+B						

$\text{Min}(c_{11}v_{11}z_{11}y1_1, c_{21}v_{11}z_{21}y1_2)$

	0	0	0	0	0		0	0	0	0	0
	48	120	0	64	16		48	0	48	24	6
	48	120		64	248		48		48	24	126

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

24	40	0	0	24
48	120	0	64	16
24	40		64	168

Min = 168

24	0	24	0	12
48	0	48	24	6
24		24	24	90

Min = 90

Total Min = 258

Case 2

k=1

24	40	0	32	24
0	0	0	0	0
24	40		32	120

0	0	0	0	0
48	120	0	64	16
48	120		64	248

24	40	0	32	24
48	120	0	64	16
24	40		32	136

Min = 120

k=2

24	0	0	12	12
0	0	0	0	0

0	0	0	0	0
48	0	48	24	6
48		48	24	126

24	0	0	12	12
48	0	48	24	6
24		48	12	102

Min = 102

Total Min = 222

Case 3

k=1

24	0	0	32	24
0	0	0	0	0

k=2

24	0	24	12	12
0	0	0	0	0
24		24	12	72

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

0	0	0	0	0
48	120	0	64	16
48	120		64	248

0	0	0	0	0
48	0	48	24	6
48		48	24	126

24	0	0	32	24
48	120	0	64	16
24	120		32	216

24	0	0	24	12
48	0	48	24	6
24		48	12	102

Min = 216

Min = 72

Total Min = 288

Case 4

k=1

0	40	0	32	24
0	0	0	0	0

0	0	0	0	0
48	120	0	64	16
48	120		64	248

0	40	0	32	24
48	120	0	64	16
48	40		32	160

Min = 160

k=2

0	0	24	12	12
0	0	0	0	0

0	0	0	0	0
48	0	48	24	6
48		48	24	126

0	0	24	12	12
48	0	48	24	6
48		24	12	102

Min = 102

Total Min = 262

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Solution :

Total Min = 222

z_{ij}	j=1	2	3	4
i= 1	1	1	0	1
2	1	1	1	1

y_{ik}	k=1	2
i= 1	1	1
2	0	1

k=1

x_{ij}	j=1	2	3	4
i= 1	1	1	0	1
2	0	0	0	0

k=2

x_{ij}	j=1	2	3	4
i= 1	1	0	0	1
2	0	0	1	0

Solution of z_{ij}

Component #1, #2 and #3 are assign to process #1.

Component #1, #2, #3 and #4 are assign to process #2.

Solution of y_{ik}

Board #1 is set up on process #1.

Board #2 is setup on process #1 and #2.

Solution of x_{ij}

Board #1: Component #1, #2 and #4 are assign to process #1.

Board #2: Component #1 and #4 are assign to process #1.

Component #3 is assign to process #2.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

4.2 APPLICATION PROBLEMS

Example 4.2 We consider problems of single machine problems.

Table 4.1 The Results of Single Machine Problems

Data Set	Number of Boards	Number of Components	Greedy Board		Variable Decomposition	
			Cost	CPU Time (Sec.)	Cost	CPU Time (Sec.)
1	10	95	92,568	< 1	53,000	3
2	10	114	142,269	< 1	76,650	5
3	30	316	869,058	1	456,600	206

Example 4.3 We consider problems of two machines.

Table 4.2 The Results of Two-Machine Problems

Data Set	Number of Boards	Number of Components	Greedy Board		Variable Decomposition	
			Cost	CPU Time (Sec.)	Cost	CPU Time (Sec.)
1	10	95	92,568	< 1	50,850	15
2	10	114	142,269	< 1	73,950	21
3	30	316	869,058	1	451,150	690

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 5

CONCLUSION AND SUGGESTION

An operation assignment problem arise from a printed circuit board assembly process was formulated as a binary integer program. The BIP was too large the solve using standard integer programming codes, but a special structure that could be exploited. By applying Variable Decomposition, the problem could be decomposed. It breaks a difficult problem into smaller and easier sub-problems. The sub-problems are then solved iteratively and their solutions coordinated in a proper way to drive the overall solution to optimality of the original problem.

From the results of examples in this research, the Variable Decomposition solutions are optimal, but CPU times are greater than Greedy Board heuristic.

The Greedy Board heuristic could be use that the machine are identical, but the Variable Decomposition is using general propose.

For the examples, running times using Borland Jbuilder3 Professional program on Pentium IV. If the program is running on an efficient CPU, it can be solve the problems are larger and faster.

Since the number of combinations for z values is still quite high for high values of N and $1 << N_i << N$. In future research, some heuristics and more smart algorithms will be developed to solve Problem (3.5) in these cases even more efficiently.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

REFERENCES

[1] Christos H. Papadimitriou and kenneth Steiglitz. **Combinatorial Optimization: Algorithms and Complexity**. Prentice-Hall, Inc. 1982.

[2] D.M. Himmelblau. **Decomposition of Large Scale Systems**. North Holland. 1973.

[3] J.C. Ammons, C.B. Olfgren and L.F. McGinnis. "A Large Scale Machine Loading Problem in Flexible Assembly." Annals of Operation Research, vol. 3, 1985. pp. 319-332.

[4] J.R. King and V. Nakornchai. "Machine-Component Group Formation in Group Technology: Review and Extension." International Journal of Production Research, vol. 20, no. 2, 1982. pp. 117-133.

[5] John W. Mamer and Andrew W. Shogan. "A Constrained Capital Budgeting Problem with Applications to Repair Kit Selection." Management Science, vol. 33, no. 6, June 1987. pp. 800-806.

[6] Leon S. Lasdon. **Optimization Theory for Large System**. MacMillan Co. 1973.

[7] M. L. Brandeau and C. A. Billington. "Design of manufacturing cells:operation assingment in printed circuit board manufacturing." Journal of Intelligent Manufacturing, vol. 2, 1991. pp. 95-106.

[8] Madan G Singh and Andre Titli. **Systems: Decomposition, Optimization and Control**. Pergaman Press. 1978.

[9] Mark S. Hillier and Margaret L. Brandeau. "Optimal component assignment and board in printed circuit board manufacturing." Operation Research, vol. 46, no. 5, September – October 1998. pp. 675-689.

[10] Stanistaw Walukiewicz. **Integer Programming**. Kluwer Academic Pubishers. 1990.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

APPENDIX

We show source code of Variable Decomposition method by Borland JBuilder3 Professional program.

```
//Title:      Your Product Name
//Version:
//Copyright:
//Author:     Your Name
//Company:    Your Company
//Description:Your description
package untitled2;

public class Application1 {
 boolean packFrame = false;
 //Construct the application
 public Application1() {
  MainFrame frame = new MainFrame();
  //Validate frames that have preset sizes
  //Pack frames that have useful preferred size info, e.g. from their layout
  if (packFrame)
   frame.pack();
  else
   frame.validate();
  //Center the window
  Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
  Dimension frameSize = frame.getSize();
  if (frameSize.height > screenSize.height)
   frameSize.height = screenSize.height;
  if (frameSize.width > screenSize.width)
   frameSize.width = screenSize.width;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  Button nextButton = new Button();
  public GridControl gridControl1 = new GridControl();
  public CijkFrame(MainFrame owner) {
//my code to goes here
    this.owner=owner;
    this.vectorCijkFrame=owner.vectorCijkFrame;
    this.vectorCijkFrame=owner.vectorCijkFrame;
    this.input_i=owner.input_i;
    this.input_j=owner.input_j;
    this.input_k=owner.input_k;
    this.cijkFrameId=owner.cijkFrameId;
    this.vectorTFrame=owner.vectorTFrame;
    this.njkFrame=owner.njkFrame;
    this.vectorCijkArray=owner.vectorCijkArray;
//    this.dup=owner.dup;
//    this.indexOrder=order;
//    this.vk=frame1.vk;
    try {
     jbInit();
    }
    catch(Exception e) {
     e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(360,330));
    this.setResizable(true);
    label1.setBackground(Color.blue);
    label1.setFont(new java.awt.Font("Dialog", 1, 28));
    label1.setForeground(Color.red);
    label1.setAlignment(1);
    label1.setText("cijk Table");
```

This content is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
this.getContentPane().setLayout(borderLayout1);
panel1.setBackground(Color.blue);
prevButton.setFont(new java.awt.Font("Dialog", 1, 12));
prevButton.setForeground(Color.black);
prevButton.setLabel("<< Prev");
prevButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  prevButton_actionPerformed(e);
 }
});
okButton.setFont(new java.awt.Font("Dialog", 1, 12));
okButton.setForeground(Color.black);
okButton.setLabel("   Ok   ");
okButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  okButton_actionPerformed(e);
 }
});
nextButton.setFont(new java.awt.Font("Dialog", 1, 12));
nextButton.setForeground(Color.black);
nextButton.setLabel("Next >>");
nextButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  nextButton_actionPerformed(e);
 }
});
this.setTitle("k = "+(getFrameId()+1));
this.addWindowListener(new java.awt.event.WindowAdapter() {
 public void windowClosing(WindowEvent e) {
  this_windowClosing(e);
 }
});
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    this.getContentPane().add(label1, BorderLayout.NORTH);
    this.getContentPane().add(panel1, BorderLayout.SOUTH);
    panel1.add(prevButton, null);
    panel1.add(okButton, null);
    panel1.add(nextButton, null);
    this.getContentPane().add(gridControl1, BorderLayout.CENTER);
//----- my code to goes here -------
    addColumns(input_j);
    addRows(input_i);
//  prevButton.disable();
//  nextButton.disable();
    if(0==getFrameId()) prevButton.disable();
    if(input_k==(getFrameId()+1)) nextButton.disable();
//   setShowButton(false);
    setOkButton(false);
    if(owner.dup==true){ // take answer YES on Duplicate Dialogbox
      prevButton.disable();
      nextButton.disable();
      okButton.enable();
    }
 //Load the stopword file and place individual words in a vector
            BufferedReader in = new BufferedReader(new FileReader("cijk.txt"));
            Vector buf = new Vector(10);
            String s;
      String sx[][] = new String[input_i][input_j];
      int row=0;
      System.out.println("\n====== rjk =====\n");
            while ((s = in.readLine()) != null){
       StringTokenizer st = new StringTokenizer(s," ");
       int col = 0;
       while(st.hasMoreTokens()){
         sx[row][col]=st.nextToken();
```

This material is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
        System.out.print(sx[row][col]);
        col++;
       }
       System.out.print("\n");
       row++;
     }
            in.close();
    gridControl1.setItems(sx);
  }
  public void addColumns(int col){
   for(int i=1;i<col;i++){
     gridControl1.addColumn();
   }
  }
  public void addRows(int row){
   for(int i=1;i<row;i++){
     gridControl1.addRow();
   }
  }
  public int getFrameId(){
   return cijkFrameId;
  }
  void prevButton_actionPerformed(ActionEvent e) {
   CijkFrame cijk=(CijkFrame)vectorCijkFrame.elementAt(getFrameId()-1);
   Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
   Dimension frameSize = cijk.getSize();
   if (frameSize.height > screenSize.height)
     frameSize.height = screenSize.height;
   if (frameSize.width > screenSize.width)
     frameSize.width = screenSize.width;
   cijk.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    cijk.setVisible(true);
    if(cijk.getFrameId()+1==input_k)
      cijk.setOkButton(true);
    else cijk.setOkButton(false);
    cijk.show();
    this.hide();
//    this.dispose();
  }
  void okButton_actionPerformed(ActionEvent e) {
    if(!isNew){
      TFrame tFrame=(TFrame)vectorTFrame.elementAt(0);
      for(int i=0;i<8;i++){
        System.out.println("TFrame is: "+tFrame.N[i]+"\n");
      }
//    YZ_table yzTable=new YZ_table(owner);
//--- show njkFrame ----
//    njkFrame=new NjkFrame(owner) ;
    // set this frame to center.
      Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
      Dimension frameSize = njkFrame.getSize();
      if (frameSize.height > screenSize.height)
        frameSize.height = screenSize.height;
      if (frameSize.width > screenSize.width)
        frameSize.width = screenSize.width;
      njkFrame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
 frameSize.height) / 2);
      njkFrame.setVisible(true);
      njkFrame.show();
//    this.hide();
//    this.dispose();
      okButton.disable();
    }else{
```

Thai Material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    YZ_table yzTable=new YZ_table(owner);
   }
  }
  void nextButton_actionPerformed(ActionEvent e) {
//   owner.vectorCijkFrame.insertElementAt(this,getFrameId());
   CijkFrame cijk=(CijkFrame)vectorCijkFrame.elementAt(getFrameId()+1);
   Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
   Dimension frameSize = cijk.getSize();
   if (frameSize.height > screenSize.height)
    frameSize.height = screenSize.height;
   if (frameSize.width > screenSize.width)
    frameSize.width = screenSize.width;
   cijk.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
   cijk.setVisible(true);
   if(cijk.getFrameId()+1 >= input_k){
    cijk.setOkButton(true);
    cijk.setNextButton(false);
    cijk.setPrevButton(true);
   }else{
    cijk.setPrevButton(true);
    cijk.setOkButton(false);
    cijk.setNextButton(true);
   }
   if(isNew){
    cijk.setNew(true);
    cijk.show_new();
    this.setNew(false);
   }
   this.hide();
   cijk.show();
   //this.hide();
```

This is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
}
void this_windowClosing(WindowEvent e) {
 this.hide();
 this.dispose();
}
public void fillGridItems(){
  String[][] cijk=(String[][])vectorCijkArray.elementAt(getFrameId());
  cijk=gridControl1.getItems();
//  for(int k=0;k<input_k;k+++){
    for(int i=0;i<input_i;i++){
     for(int j=0;j<input_j;j++){
      cijk[i][j]=String.valueOf(Integer.parseInt(owner.vjkArray[j][getFrameId()])
                  *(Integer.parseInt(cijk[i][j])));
     }
    }
//  }//for(k);
    gridControl1.setItems(cijk);
    owner.vectorCijkArray.insertElementAt(cijk,getFrameId());
 }
 public void show_new() {
  fillGridItems();
//  setShowButton(false);
 }
 public void setOkButton(boolean b){
  if(b)okButton.enable();
  else okButton.disable();
 }
 public void setPrevButton(boolean b){
  if(b)prevButton.enable();
  else prevButton.disable();
 }
 public void setNextButton(boolean b){
```

This material is recommended for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    if(b)nextButton.enable();
    else nextButton.disable();
  }
  public void setTextTitle(String s){
    this.setTitle(s);
  }
  public void setNew(boolean n){
    isNew=n;
  }
  public String[][] getGridItems(){
    return gridControl1.getItems();
  }
}

public class DkFrame extends JFrame {
//--- my variable to goes here ---
  private MainFrame owner;
  private Vector vectorDkFrame;
  private Vector vField=new Vector();
  private int input_i,input_j,input_k;
  public  int[] N=new int[8];
  private VjkFrame vjkFrame;
  private SikFrame sikFrame;
  public  int frameId=0;
  Button prevButton = new Button();
  Button okButton = new Button();
  Button nextButton = new Button();
  Label label1 = new Label();
  Label n2Label = new Label();
  TextField n1Field = new TextField();
  TextField n2Field = new TextField();
  TextField n3Field = new TextField();
```

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

```
TextField n4Field = new TextField();
TextField n5Field = new TextField();
TextField n6Field = new TextField();
TextField n7Field = new TextField();
TextField n8Field = new TextField();
Label n1Label = new Label();
Label n3Label = new Label();
Label n4Label = new Label();
Label n5Label = new Label();
Label n6Label = new Label();
Label n7Label = new Label();
Label n8Label = new Label();
public DkFrame(MainFrame owner) {
//--------- my code to goes here ---------
  this.owner=owner;
  this.vjkFrame=owner.vjkFrame;
  this.vectorDkFrame=owner.vectorDkFrame;
  this.input_i=owner.input_i;
  this.input_j=owner.input_j;
  this.input_k=owner.input_k;
  this.frameId=owner.dkFrameId;
  this.sikFrame=owner.sikFrame;
  try {
   jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
}
private void jbInit() throws Exception {
  this.setSize(new Dimension(360,330));
  this.setTitle("dk Table");
```

This thesis is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
this.addWindowListener(new java.awt.event.WindowAdapter() {
 public void windowClosing(WindowEvent e) {
  this_windowClosing(e);
 }
});
prevButton.setBackground(Color.lightGray);
prevButton.setBounds(new Rectangle(84, 233, 60, 30));
prevButton.setFont(new java.awt.Font("Dialog", 1, 12));
prevButton.setForeground(Color.black);
prevButton.setLabel("<< Prev");
prevButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  prevButton_actionPerformed(e);
 }
});
this.getContentPane().setLayout(null);
okButton.setBackground(Color.lightGray);
okButton.setBounds(new Rectangle(152, 232, 44, 30));
okButton.setFont(new java.awt.Font("Dialog", 1, 12));
okButton.setForeground(Color.black);
okButton.setLabel("Ok");
okButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  okButton_actionPerformed(e);
 }
});
nextButton.setBackground(Color.lightGray);
nextButton.setBounds(new Rectangle(205, 232, 63, 30));
nextButton.setFont(new java.awt.Font("Dialog", 1, 12));
nextButton.setForeground(Color.black);
nextButton.setLabel("Next >>");
nextButton.addActionListener(new java.awt.event.ActionListener() {
```

This material is for educational purposes only. Not for commercial use.
Forbidden to modify the content, and cite the document when use.

```
 public void actionPerformed(ActionEvent e) {
   nextButton_actionPerformed(e);
 }
});
label1.setBounds(new Rectangle(89, 12, 137, 29));
label1.setFont(new java.awt.Font("Dialog", 1, 28));
label1.setForeground(Color.red);
label1.setAlignment(1);
label1.setText("Input  dk");
n2Label.setBounds(new Rectangle(6, 99, 49, 26));
n2Label.setFont(new java.awt.Font("Dialog", 1, 16));
n2Label.setForeground(Color.red);
n2Label.setText("d2");
n1Field.setBounds(new Rectangle(60, 72, 71, 18));
n1Field.setEditable(false);
n2Field.setBounds(new Rectangle(59, 103, 71, 19));
n2Field.setEditable(false);
n3Field.setBounds(new Rectangle(59, 135, 72, 19));
n3Field.setEditable(false);
n4Field.setBounds(new Rectangle(58, 166, 73, 19));
n4Field.setEditable(false);
n5Field.setBounds(new Rectangle(201, 71, 73, 20));
n5Field.setEditable(false);
n6Field.setBounds(new Rectangle(201, 102, 74, 20));
n6Field.setEditable(false);
n7Field.setBounds(new Rectangle(200, 135, 73, 20));
n7Field.setEditable(false);
n8Field.setBounds(new Rectangle(199, 167, 73, 21));
n8Field.setEditable(false);
n1Label.setText("d"+(2+(getFrameId()*8)));
n1Label.setText("d1");
n1Label.setForeground(Color.red);
```

This thesis is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
n1Label.setFont(new java.awt.Font("Dialog", 1, 16));
n1Label.setBounds(new Rectangle(5, 66, 49, 26));
n3Label.setText("d"+(2+(getFrameId()*8)));
n3Label.setText("d3");
n3Label.setForeground(Color.red);
n3Label.setFont(new java.awt.Font("Dialog", 1, 16));
n3Label.setBounds(new Rectangle(5, 131, 49, 26));
n4Label.setText("d"+(2+(getFrameId()*8)));
n4Label.setText("d4");
n4Label.setForeground(Color.red);
n4Label.setFont(new java.awt.Font("Dialog", 1, 16));
n4Label.setBounds(new Rectangle(6, 162, 49, 26));
n5Label.setText("d"+(2+(getFrameId()*8)));
n5Label.setText("d5");
n5Label.setForeground(Color.red);
n5Label.setFont(new java.awt.Font("Dialog", 1, 16));
n5Label.setBounds(new Rectangle(149, 66, 49, 26));
n6Label.setText("d"+(2+(getFrameId()*8)));
n6Label.setText("d6");
n6Label.setForeground(Color.red);
n6Label.setFont(new java.awt.Font("Dialog", 1, 16));
n6Label.setBounds(new Rectangle(149, 97, 49, 26));
n7Label.setText("d"+(2+(getFrameId()*8)));
n7Label.setText("d7");
n7Label.setForeground(Color.red);
n7Label.setFont(new java.awt.Font("Dialog", 1, 16));
n7Label.setBounds(new Rectangle(149, 131, 49, 26));
n8Label.setText("d"+(2+(getFrameId()*8)));
n8Label.setText("d8");
n8Label.setForeground(Color.red);
n8Label.setFont(new java.awt.Font("Dialog", 1, 16));
n8Label.setBounds(new Rectangle(148, 164, 49, 26));
```

This thesis is in the Digital Collection of King Mongkut's University of Technology Ladkrabang (KMITL). Not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    this.getContentPane().add(prevButton, null);
    this.getContentPane().add(nextButton, null);
    this.getContentPane().add(label1, null);
    this.getContentPane().add(n1Field, null);
    this.getContentPane().add(n2Field, null);
    this.getContentPane().add(n3Field, null);
    this.getContentPane().add(n4Field, null);
    this.getContentPane().add(n5Field, null);
    this.getContentPane().add(n6Field, null);
    this.getContentPane().add(n7Field, null);
    this.getContentPane().add(n8Field, null);
    this.getContentPane().add(okButton, null);
    this.getContentPane().add(n2Label, null);
    this.getContentPane().add(n1Label, null);
    this.getContentPane().add(n5Label, null);
    this.getContentPane().add(n6Label, null);
    this.getContentPane().add(n8Label, null);
    this.getContentPane().add(n4Label, null);
    this.getContentPane().add(n3Label, null);
    this.getContentPane().add(n7Label, null);
//---- my code to goes here ----
    vField.addElement(n1Field);
    vField.addElement(n2Field);
    vField.addElement(n3Field);
    vField.addElement(n4Field);
    vField.addElement(n5Field);
    vField.addElement(n6Field);
    vField.addElement(n7Field);
    vField.addElement(n8Field);
    n1Label.setText("d"+(1+(getFrameId()*8)));
    n2Label.setText("d"+(2+(getFrameId()*8)));
    n3Label.setText("d"+(3+(getFrameId()*8)));
```

This document is for study purpose only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    n4Label.setText("d"+(4+(getFrameId()*8)));
    n5Label.setText("d"+(5+(getFrameId()*8)));
    n6Label.setText("d"+(6+(getFrameId()*8)));
    n7Label.setText("d"+(7+(getFrameId()*8)));
    n8Label.setText("d"+(8+(getFrameId()*8)));
    int i=getFrameId()*8;
    int size=0;
    TextField t;
//ystem.out.println("enter while loop -> i="+i+" -> input_i="+input_i+"\n");
    while((i<input_k)&&(size<vField.size())){
      t=(TextField)vField.elementAt(size);
      t.setEditable(true);
//System.out.println(t.getText()+i+" t\n");
      i++;
      size++;
    }
//System.out.println("exit while loop\n");
    if(0==getFrameId()){
      prevButton.disable();
      nextButton.enable();
      okButton.disable();
    }
    if(input_k-getFrameId()*8<=8){
      nextButton.disable();
      okButton.enable();
//      prevButton.enable();
    }
    for(int ii=0;ii<8;ii++){
      N[ii]=0;
    }
  }
  public void setFrameId(int id){
```

This material is used for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  frameId=id;
}
public int getFrameId(){
  return frameId;
}
void nextButton_actionPerformed(ActionEvent e) {
  try{
    this.N[0]=Integer.parseInt(n1Field.getText());
    this.N[1]=Integer.parseInt(n2Field.getText());
    this.N[2]=Integer.parseInt(n3Field.getText());
    this.N[3]=Integer.parseInt(n4Field.getText());
    this.N[4]=Integer.parseInt(n5Field.getText());
    this.N[5]=Integer.parseInt(n6Field.getText());
    this.N[6]=Integer.parseInt(n7Field.getText());
    this.N[7]=Integer.parseInt(n8Field.getText());
  }catch(NumberFormatException ee){}
  DkFrame dkFrame=(DkFrame)vectorDkFrame.elementAt(getFrameId()+1);  //next element of
TFrmae
//System.out.println("frameId="+getFrameId());
  Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
  Dimension frameSize = dkFrame/*frame*/.getSize();
  if (frameSize.height > screenSize.height)
    frameSize.height = screenSize.height;
  if (frameSize.width > screenSize.width)
    frameSize.width = screenSize.width;
  dkFrame/*frame*/.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
  dkFrame/*frame*/.setVisible(true);
  dkFrame.show();
  this.hide();
  this.dispose();
}
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
 void okButton_actionPerformed(ActionEvent e) {
  try{
    this.N[0]=Integer.parseInt(n1Field.getText());
    this.N[1]=Integer.parseInt(n2Field.getText());
    this.N[2]=Integer.parseInt(n3Field.getText());
    this.N[3]=Integer.parseInt(n4Field.getText());
    this.N[4]=Integer.parseInt(n5Field.getText());
    this.N[5]=Integer.parseInt(n6Field.getText());
    this.N[6]=Integer.parseInt(n7Field.getText());
    this.N[7]=Integer.parseInt(n8Field.getText());
  }catch(NumberFormatException ee){}
  //----- set VjkFrame alignment to center -------
//   this.hide();
  Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
  Dimension frameSize = owner.vjkFrame.getSize();//frame.getSize();
  if (frameSize.height > screenSize.height)
    frameSize.height = screenSize.height;
  if (frameSize.width > screenSize.width)
    frameSize.width = screenSize.width;
  owner.vjkFrame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
  owner.vjkFrame.setVisible(true);
  owner.vjkFrame.fillGridItems();
  owner.vjkFrame.show();
//   owner.sikFrame.setShowButton(true);
  this.hide();
 //  this.dispose();
  }
  void prevButton_actionPerformed(ActionEvent e) {
   DkFrame dkFrame=(DkFrame)vectorDkFrame.elementAt(getFrameId()-1);
   dkFrame.show();
   this.hide();
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  this.dispose();
}
public int getNi(int i){
  if(i<0)
    return -2; // underflow
  else if(i<8){
    return N[i];
  }else
    return -1; // overflow
 }
 void this_windowClosing(WindowEvent e) {
   this.hide();
   this.dispose();
 }
 public int[] getItems(){
   return N;
 }
}

public class Error_dialog extends JDialog {
//-- my variable to goes here ---
  MainFrame owner;
  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();
  JButton yesButton = new JButton();
  Border border1;
  JPanel jPanel1 = new JPanel();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  GridLayout gridLayout1 = new GridLayout();
  XYLayout xYLayout1 = new XYLayout();
  Label label1 = new Label();
  String errorMsg;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
public Error_dialog(Frame frame, String title, boolean modal,MainFrame owner,String msg) {
  super(frame, title, modal);
  this.owner=owner;
  this.errorMsg=msg;
  try {
    jbInit();
  }
  catch (Exception e) {
    e.printStackTrace();
  }
  pack();
}
private void jbInit() throws Exception {
  this.setSize(new Dimension(360,330));
  border1 = BorderFactory.createRaisedBevelBorder();
  jPanel1.setLayout(gridLayout1);
  panel2.setBorder(border1);
  panel2.setLayout(xYLayout1);
  yesButton.setText("Yes");
  yesButton.addActionListener(new Error_dialog_okButton_actionAdapter(this));
  gridLayout1.setHgap(4);
  this.addWindowListener(new Error_this_windowAdapter(this));
  panel1.setLayout(gridBagLayout1);
  label1.setFont(new java.awt.Font("Dialog", 1, 26));
  label1.setForeground(Color.red);
  label1.setAlignment(1);
  label1.setText(errorMsg);
  panel1.add(jPanel1, new GridBagConstraints(0, 1, 1, 1, 1.0, 0.0
        ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(4, 8, 4, 8), 0, 0));
  jPanel1.add(yesButton, null);
  panel1.add(panel2, new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0
        ,GridBagConstraints.EAST, GridBagConstraints.NONE, new Insets(2, 7, 0, 13), 4, 44));
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    panel2.add(label1, new XYConstraints(6, 34, 290, -1));
    getContentPane().add(panel1);
  }
  // OK
  void yesButton_actionPerformed(ActionEvent e) {
    owner.isError=true;
    dispose();
  }
  // Cancel
  void this_windowClosing(WindowEvent e) {
//    dispose();
  }
  public void setMsg(String msg){
    setModal(false);
    label1.setText(msg);
    setModal(true);
  }
}
class Error_dialog_okButton_actionAdapter implements ActionListener {
  Error_dialog adaptee;
  Error_dialog_okButton_actionAdapter(Error_dialog adaptee) {
    this.adaptee = adaptee;
  }
  public void actionPerformed(ActionEvent e) {
    adaptee.yesButton_actionPerformed(e);
  }
}
class Error_this_windowAdapter extends WindowAdapter {
  Error_dialog adaptee;
  Error_this_windowAdapter(Error_dialog adaptee) {
    this.adaptee = adaptee;
  }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  public void windowClosing(WindowEvent e) {
    adaptee.this_windowClosing(e);
  }
}

public class MainFrame extends JFrame {
//--------- my variable to goes here
  public Vector vectorTFrame=new Vector();
  public Vector vectorCijkFrame=new Vector();
  public Vector vectorDkFrame=new Vector();
  public Vector vectorCijkArray=new Vector();
  public RjkFrame rjkFrame;
  public SikFrame sikFrame;
  public CijkFrame cijkFrame;
  public NjkFrame njkFrame;
  public VjkFrame vjkFrame;
  public String[][] rjkArray;
  public String[][] sikArray;
  public String[][] njkArray;
  public String[][] vjkArray;
  public int input_i,input_j,input_k;
  public static boolean dup;
  public static boolean isError=false;
 //  TFrame tFrame;
  public static int tFrameId=0;
  public static int cijkFrameId=0;
  public static int dkFrameId=0;
  JMenuBar menuBar1 = new JMenuBar();
  JMenu menuFile = new JMenu();
  JMenuItem menuFileExit = new JMenuItem();
  JMenu menuHelp = new JMenu();
  JMenuItem menuHelpAbout = new JMenuItem();
```

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

```
JToolBar toolBar = new JToolBar();
JButton jButton1 = new JButton();
JButton jButton2 = new JButton();
JButton jButton3 = new JButton();
ImageIcon image1;
ImageIcon image2;
ImageIcon image3;
JLabel statusBar = new JLabel();
Label label1 = new Label();
Label label2 = new Label();
TextField textField_i = new TextField();
Label label3 = new Label();
TextField textField_j = new TextField();
Label label4 = new Label();
TextField textField_k = new TextField();
Button okButton = new Button();
Button cancelButton = new Button();
Label label5 = new Label();
//Construct the frame
public MainFrame() {
 enableEvents(AWTEvent.WINDOW_EVENT_MASK);
 try {
  jbInit();
 }
 catch(Exception e) {
  e.printStackTrace();
 }
}
//Component initialization
private void jbInit() throws Exception  {
 this.setSize(new Dimension(360,330));
 this.setFont(new java.awt.Font("Dialog", 1, 20));
```

Thesis is reserved for reading only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
this.setResizable(true);
image1 = new ImageIcon(untitled2.MainFrame.class.getResource("openFile.gif"));
image2 = new ImageIcon(untitled2.MainFrame.class.getResource("closeFile.gif"));
image3 = new ImageIcon(untitled2.MainFrame.class.getResource("help.gif"));
this.getContentPane().setLayout(null);
this.setTitle("NungNing");
this.addWindowListener(new java.awt.event.WindowAdapter() {
 public void windowClosing(WindowEvent e) {
  this_windowClosing(e);
 }
});
statusBar.setText(" ");
statusBar.setBounds(new Rectangle(0, 279, 392, 16));
menuFile.setText("File");
menuFileExit.setText("Exit");
menuFileExit.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  fileExit_actionPerformed(e);
 }
});
menuHelp.setText("Help");
menuHelpAbout.setText("About");
menuHelpAbout.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  helpAbout_actionPerformed(e);
 }
});
jButton1.setIcon(image1);
jButton1.setToolTipText("Open File");
jButton2.setIcon(image2);
jButton2.setToolTipText("Close File");
jButton3.setIcon(image3);
```

This material is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
jButton3.setToolTipText("Help");
label1.setBounds(new Rectangle(96, 40, 173, 36));
label1.setFont(new java.awt.Font("Dialog", 1, 28));
label1.setForeground(Color.red);
label1.setAlignment(1);
label1.setText("Enter Input");
label2.setBounds(new Rectangle(-60, 0, 54, 31));
label2.setFont(new java.awt.Font("Dialog", 1, 20));
label2.setForeground(Color.red);
label2.setAlignment(1);
label2.setText("Input  i");
label3.setBounds(new Rectangle(62, 130, 63, 24));
label3.setFont(new java.awt.Font("Dialog", 1, 16));
label3.setForeground(Color.blue);
label3.setAlignment(1);
label3.setText("Input  j");
label4.setBounds(new Rectangle(64, 163, 58, 36));
label4.setFont(new java.awt.Font("Dialog", 1, 16));
label4.setForeground(Color.blue);
label4.setText("Input  k");
okButton.setBounds(new Rectangle(120, 224, 70, 32));
okButton.setFont(new java.awt.Font("Dialog", 1, 12));
okButton.setLabel("Ok");
okButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
    okButton_actionPerformed(e);
  }
});
cancelButton.setBounds(new Rectangle(198, 224, 73, 33));
cancelButton.setFont(new java.awt.Font("Dialog", 1, 12));
cancelButton.setLabel("Cancel");
cancelButton.addActionListener(new java.awt.event.ActionListener() {
```

This material is protected by copyright. Non-commercial use.
Forbidden to modify the content, and cite the document when use.

```
  public void actionPerformed(ActionEvent e) {
    cancelButton_actionPerformed(e);
  }
});
label5.setBounds(new Rectangle(63, 86, 60, 38));
label5.setFont(new java.awt.Font("Dialog", 1, 16));
label5.setForeground(Color.blue);
label5.setAlignment(1);
label5.setText("Input  i");
textField_k.setBounds(new Rectangle(137, 169, 129, 24));
textField_k.setText("10");
textField_k.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
    textField_k_actionPerformed(e);
  }
});
textField_j.setBounds(new Rectangle(137, 132, 127, 24));
textField_j.setText("13");
textField_i.setBounds(new Rectangle(136, 94, 128, 21));
textField_i.setText("3");
toolBar.setBounds(new Rectangle(0, 0, 392, 27));
toolBar.add(jButton1);
toolBar.add(jButton2);
toolBar.add(jButton3);
menuFile.add(menuFileExit);
menuHelp.add(menuHelpAbout);
menuBar1.add(menuFile);
menuBar1.add(menuHelp);
this.setJMenuBar(menuBar1);
this.getContentPane().add(toolBar, null);
this.getContentPane().add(statusBar, null);
this.getContentPane().add(label1, null);
```

This thesis is for personal use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
 this.getContentPane().add(label2, null);
 this.getContentPane().add(label5, null);
 this.getContentPane().add(textField_i, null);
 this.getContentPane().add(textField_j, null);
 this.getContentPane().add(textField_k, null);
 this.getContentPane().add(okButton, null);
 this.getContentPane().add(cancelButton, null);
 this.getContentPane().add(label4, null);
 this.getContentPane().add(label3, null);
}
//File | Exit action performed
public void fileExit_actionPerformed(ActionEvent e) {
  if(rjkFrame!=null)rjkFrame.dispose();
  if(sikFrame!=null)sikFrame.dispose();
  if(vjkFrame!=null)vjkFrame.dispose();
  System.exit(0);
}
//Help | About action performed
public void helpAbout_actionPerformed(ActionEvent e) {
 // MainFrame_AboutBox dlg = new MainFrame_AboutBox(this);
  TextArea text=new TextArea();
  Font font1=new Font("TimesRoman",Font.BOLD,16);
  text.setFont(font1);
  text.setText  ("Notation\n");
  text.appendText("=========\n\n");
  Font font2=new Font("TimesRoman",Font.PLAIN,14);
  text.setFont(font2);
  text.appendText("\t  i\t = number of processes.\n"+
                  "\t  j\t = number of components.\n"+
                  "\t  k\t = number of boards.\n"+
                  "\t Ni\t = total number of dufferent types of components that can be assigned to
process  i .\n"+
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
            "\t dk\t = expected demand for board k during the time horizon.\n"+
            "\tCij\t = cost of insertion of component j on process i .\n"+
            "\tSik\t = set up cost for board k on process i .\n"+
            "\trjk\t = 1 if component j is used in board k, 0 otherwise.\n"+
            "\tnjk\t = number of component j used in board k.\n"+
            "\tvjk\t = expected number of component j used in board k during the time horizon(=
dk*njk).\n\n");
   text.setFont(font1);
   text.appendText("Solution\n");
   text.appendText("========\n\n");
   text.setFont(font2);
   text.appendText("\t Zij\t = 1 if component j is assign to process i, 0 otherwise.\n"+
            "\t Yik\t = 1 if board k is set up on process i, 0 otherwise.\n"+
            "\tXijk\t = 1 if component j of board k is assigned to process i, 0 otherwise.\n");
   MainFrame_AboutBox dlg = new MainFrame_AboutBox(this,"Help",true,this,text);
   Dimension dlgSize = dlg.getPreferredSize();
   Dimension frmSize = getSize();
   Point loc = getLocation();
   dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height - dlgSize.height) /
2 + loc.y);
   dlg.setModal(true);
   dlg.show();
  }
  //Overridden so we can exit on System Close
  protected void processWindowEvent(WindowEvent e) {
   super.processWindowEvent(e);
   if(e.getID() == WindowEvent.WINDOW_CLOSING) {
    fileExit_actionPerformed(null);
   }
  }
  void okButton_actionPerformed(ActionEvent e) {
   try{
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    input_i=Integer.parseInt(textField_i.getText());
    input_j=Integer.parseInt(textField_j.getText());
    input_k=Integer.parseInt(textField_k.getText());
    isError=false;
   }catch(NumberFormatException ee){
    String msg="You must be fill the number all fieds.";
    Error_dialog err=new Error_dialog(this,"Error message",true,this,msg);
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = err/*frame*/.getSize();
    if (frameSize.height > screenSize.height)
      frameSize.height = screenSize.height;
    if (frameSize.width > screenSize.width)
      frameSize.width = screenSize.width;
    err/*frame*/.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
     err/*frame*/.setVisible(true);
     err.setMsg("You must be fill all fields");
//     err.setSize(new Dimension(200,200));
//     err.setMsg("You must be fill all fields");
//     err.show();
   }
   if(!isError){
   rjkArray=new String[input_j][input_k];
   sikArray=new String[input_i][input_k];
   njkArray=new String[input_j][input_k];
   vjkArray=new String[input_j][input_k];
   String[][] d;
   for(int i=0;i<input_k;i++){
     d=new String[input_i][input_j];
     vectorCijkArray.addElement(d);
   }
 //---------- create TFrame ----------
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
int n=input_i;
TFrame tFrame;
tFrameId=0; //initialize
do{
 tFrame=new TFrame(this);
 vectorTFrame.addElement(tFrame);
 n=n-8;
 tFrameId++;
}while((n/8)>0);
if(n>0){ //last element
 tFrame=new TFrame(this);
 vectorTFrame.addElement(tFrame);
}
//---------- create DkFrame ----------
int nn=input_k;
DkFrame dkFrame;
dkFrameId=0; //initialize
do{
 dkFrame=new DkFrame(this);
 vectorDkFrame.addElement(dkFrame);
 nn=nn-8;
 dkFrameId++;
}while((nn/8)>0);
if(nn>0){ //last element
 dkFrame=new DkFrame(this);
 vectorDkFrame.addElement(dkFrame);
}
//---------- creating RjkFrame & SikFrame ----------
rjkFrame=new RjkFrame(this);
sikFrame=new SikFrame(this);
vjkFrame=new VjkFrame(this);
njkFrame=new NjkFrame(this) ;
```

This thesis is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
//   rjkFrame.show();
//---------- alignment to center ----------------------
    tFrame=(TFrame)vectorTFrame.elementAt(0);  //first element of TFrame
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = tFrame/*frame*/.getSize();
    if (frameSize.height > screenSize.height)
      frameSize.height = screenSize.height;
    if (frameSize.width > screenSize.width)
      frameSize.width = screenSize.width;
    tFrame/*frame*/.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    tFrame/*frame*/.setVisible(true);
    tFrame.show();
    okButton.disable();
   }
  }
  void cancelButton_actionPerformed(ActionEvent e) {
    textField_i.setText(" ");
    textField_j.setText(" ");
    textField_k.setText(" ");
// -----
  int x[]=new int[10];
// -----
  }
  void this_windowClosing(WindowEvent e) {
    if(njkFrame!=null){
      njkFrame.hide();
      njkFrame.dispose();
    }
    if(sikFrame!=null){
      sikFrame.hide();
      sikFrame.dispose();
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  }
  if(rjkFrame!=null){
    rjkFrame.hide();
    rjkFrame.dispose();
  }
  if(vjkFrame!=null){
    vjkFrame.hide();
    vjkFrame.dispose();
  }
  this.hide();
  this.dispose();
  System.exit(0);
 }
 void textField_k_actionPerformed(ActionEvent e) {
 }
}

public class MainFrame_AboutBox extends JDialog {
//-- my variable to goes here ---
  MainFrame owner;
  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();
  JButton yesButton = new JButton();
  Border border1;
  JPanel jPanel1 = new JPanel();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  GridLayout gridLayout1 = new GridLayout();
  XYLayout xYLayout1 = new XYLayout();
  Label label1 = new Label();
  String errorMsg;
  TextArea text;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
public MainFrame_AboutBox(Frame frame, String title, boolean modal,MainFrame
owner,TextArea text) {
  super(frame, title, modal);
  this.owner=owner;
  //this.errorMsg=msg;
  this.text=text;
  try {
   jbInit();
  }
  catch (Exception e) {
   e.printStackTrace();
  }
  pack();
 }
 private void jbInit() throws Exception {
  this.setSize(new Dimension(450,350));
  border1 = BorderFactory.createRaisedBevelBorder();
  jPanel1.setLayout(gridLayout1);
  panel2.setBorder(border1);
  panel2.setLayout(xYLayout1);
  yesButton.setText("Yes");
  yesButton.addActionListener(new MainFrame_AboutBox_okButton_actionAdapter(this));
  gridLayout1.setHgap(4);
  this.addWindowListener(new MainFrame_this_windowAdapter(this));
  panel1.setLayout(gridBagLayout1);
  panel1.add(jPanel1, new GridBagConstraints(0, 1, 1, 1, 1.0, 0.0
      ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(4, 8, 4, 8), 0, 0));
  jPanel1.add(yesButton, null);
  panel1.add(panel2, new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0
      ,GridBagConstraints.EAST, GridBagConstraints.NONE, new Insets(2, 7, 0, 13), 4, 44));
  panel2.add(text/*label1*/, new XYConstraints(6, 34, 290, -1));
  getContentPane().add(panel1);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
}
 // OK
 void yesButton_actionPerformed(ActionEvent e) {
  owner.isError=true;
  dispose();
 }
 // Cancel
 void this_windowClosing(WindowEvent e) {
//   dispose();
 }
 public void setMsg(String msg){
  setModal(false);
  label1.setText(msg);
  setModal(true);
 }
}
class MainFrame_AboutBox_okButton_actionAdapter implements ActionListener {
 MainFrame_AboutBox adaptee;
 MainFrame_AboutBox_okButton_actionAdapter(MainFrame_AboutBox adaptee) {
  this.adaptee = adaptee;
 }
 public void actionPerformed(ActionEvent e) {
  adaptee.yesButton_actionPerformed(e);
 }
}
class MainFrame_this_windowAdapter extends WindowAdapter {
 MainFrame_AboutBox adaptee;
 MainFrame_this_windowAdapter(MainFrame_AboutBox adaptee) {
  this.adaptee = adaptee;
 }
 public void windowClosing(WindowEvent e) {
  adaptee.this_windowClosing(e);
```

This thesis is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
 }
}

public class NjkFrame extends JFrame {
 Label label1 = new Label();
 public GridControl gridControl1 = new GridControl();
 Button ok_njkButton = new Button();
 //---- my variable -----
 private int input_j;
 private int input_k;
 private MainFrame owner;
 private Vector vectorDkFrame;
 public NjkFrame(MainFrame owner) {
  this.owner=owner;
  this.input_j=owner.input_j;
  this.input_k=owner.input_k;
  this.vectorDkFrame=owner.vectorDkFrame;
//  items=new String[input_j][input_k];
//  objectArray=new Object[input_j][input_k];
  try {
   jbInit();
  }
  catch(Exception e) {
   e.printStackTrace();
  }
 }
 private void jbInit() throws Exception {
  this.setSize(new Dimension(360,330));
  this.setResizable(true);
  this.setFont(new java.awt.Font("Dialog", 1, 20));
  this.setResizable(true);
  label1.setBackground(Color.lightGray);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
label1.setBounds(new Rectangle(98, 6, 184, 35));
label1.setFont(new java.awt.Font("Dialog", 1, 28));
label1.setForeground(Color.red);
label1.setAlignment(1);
label1.setText("njk Table");
this.getContentPane().setLayout(null);
this.setFont(new java.awt.Font("Dialog", 0, 18));
this.setTitle("Njk Frame");
this.addWindowListener(new java.awt.event.WindowAdapter() {
  public void windowClosing(WindowEvent e) {
    this_windowClosing(e);
  }
});
ok_njkButton.setBounds(new Rectangle(155, 262, 44, 32));
ok_njkButton.setFont(new java.awt.Font("Dialog", 1, 16));
ok_njkButton.setName("OK");
ok_njkButton.setLabel("Ok");
ok_njkButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ok_njkButton_actionPerformed(e);
  }
});
gridControl1.setBounds(new Rectangle(20, 49, 304, 188));
this.getContentPane().add(label1, null);
this.getContentPane().add(gridControl1, null);
this.getContentPane().add(ok_njkButton, null);
//---- my code to goes here ------
  addColumns(input_k);
  addRows(input_j);
//Load the stopword file and place individual words in a vector
        BufferedReader in = new BufferedReader(new FileReader("njk.txt"));
        Vector buf = new Vector(10);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
            String s;
   String sx[][] = new String[input_j][input_k];
   int row=0;
   System.out.println("\n====== rjk ======\n");
            while ((s = in.readLine()) != null){
    StringTokenizer st = new StringTokenizer(s," ");
    int col = 0;
    while(st.hasMoreTokens()){
     sx[row][col]=st.nextToken();
     System.out.print(sx[row][col]);
     col++;
    }
    System.out.print("\n");
    row++;
   }
            in.close();
   gridControl1.setItems(sx);
 }
 public void addColumns(int col){
  for(int i=2;i<=col;i++){
   gridControl1.addColumn();
   gridControl1.setColumnCaptions(new String[i]);
  }
 }
 public void addRows(int row){
  for(int i=2;i<=row;i++){
   gridControl1.addRow();
//    gridControl1.setRowCaptions(new String[i]);
  }
 }
// Ok Button action perform here.
 void ok_njkButton_actionPerformed(ActionEvent e) {
```

This content is protected and may not be shared, uploaded, or distributed. Commercial use is not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  for(int i=0;i<input_j;i++){
    for(int k=0;k<input_k;k++){
    }
  }
  owner.njkArray=getGridItems();//update on owner;
  DkFrame dkFrame=(DkFrame)vectorDkFrame.elementAt(0);  //first element of TFrame
  Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
  Dimension frameSize = dkFrame/*frame*/.getSize();
  if (frameSize.height > screenSize.height)
    frameSize.height = screenSize.height;
  if (frameSize.width > screenSize.width)
    frameSize.width = screenSize.width;
  dkFrame/*frame*/.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
  dkFrame/*frame*/.setVisible(true);
  dkFrame.show();
  this.hide();
  this.dispose();
 }
 void this_windowClosing(WindowEvent e) {
  this.hide();
  this.dispose();
 }
 public synchronized String[][] getGridItems(){
  return gridControl1.getItems();
 }
}

public class Report extends JFrame {
 BorderLayout borderLayout1 = new BorderLayout();
 Panel north = new Panel();
 TextField total = new TextField();
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
TextField time = new TextField();
Label label1 = new Label();
Label label2 = new Label();
Panel panel1 = new Panel();
Button button1 = new Button();
TextArea display = new TextArea();//"",20,50,TextArea.SCROLLBARS_VERTICAL_ONLY);
private YZ_table yz_table;
public int input_i,input_j,input_k;
public Report(YZ_table owner) {
  this.yz_table=owner;
  this.input_i=owner.input_i;
  this.input_j=owner.input_j;
  this.input_k=owner.input_k;
  try {
    jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
}
private void jbInit() throws Exception {
  this.setSize(new Dimension(825, 330));
  this.setResizable(true);
  this.getContentPane().setLayout(borderLayout1);
  label1.setFont(new java.awt.Font("Dialog", 1, 16));
  label1.setText("Total cost :");
  label2.setFont(new java.awt.Font("Dialog", 1, 16));
  label2.setText("Begin:");
  time.setColumns(25);
  total.setColumns(10);
  button1.setFont(new java.awt.Font("Dialog", 1, 16));
  button1.setForeground(Color.red);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
button1.setLabel("  Ok  ");
button1.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  button1_actionPerformed(e);
 }
});
jLabel1.setFont(new java.awt.Font("Dialog", 1, 16));
jLabel1.setText("Terminate:");
panel1.setLayout(flowLayout1);
timeEnd.setText(yz_table.stime);
timeEnd.setColumns(25);
this.getContentPane().add(north, BorderLayout.NORTH);
north.add(label1, null);
north.add(total, null);
this.getContentPane().add(panel1, BorderLayout.SOUTH);
panel1.add(label2, null);
panel1.add(time, null);
panel1.add(button1, null);
panel1.add(jLabel1, null);
panel1.add(timeEnd, null);
this.getContentPane().add(display, BorderLayout.CENTER);
time.setText(yz_table.stime);
timeEnd.setText(yz_table.etime);
total.setText(yz_table.getCost()+".");
showZ();
//System.out.println("Check point 1.....");
showY();
//System.out.println("Check point 2.....");
showX();
//System.out.println("Check point 3.....");
show();
//System.out.println("Check point 4.....");
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
}
public void showZ(){
  int[][] minz=yz_table.getMinZarray();
       Font font1=new Font("TimesRoman",Font.BOLD,20);
       Font font2=new Font("TimesRoman",Font.BOLD,15);
       display.setFont(font1);
       display.appendText("\n      Z table       \n");
       display.appendText("   ==========   \n\n");
       display.setFont(font2);
  for(int i=0;i<input_i;i++){
     for(int j=0;j<input_j;j++){
       int value=minz[i][j];
       String s=String.valueOf(value);
       display.appendText(" [ "+s+"]");
     }//j
     display.appendText("\n");
   }//i
}
  public void showY(){
       Font font1=new Font("TimesRoman",Font.BOLD,20);
       Font font2=new Font("TimesRoman",Font.BOLD,15);
       display.setFont(font1);
       display.appendText ("\n      Y table       \n");
       display.appendText("   ==========   \n\n");
       display.setFont(font2);
     String[][] yin=yz_table.getYin();
     for(int i=0;i<input_i;i++){
       for(int kk=0;kk<input_k;kk++){
         String s=String.valueOf(yin[i][kk]);
         display.appendText("[ "+s+"]");
       }
       display.appendText("\n");
```

This material is provided for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    }
  }
  public void showX(){
       Font font1=new Font("TimesRoman",Font.BOLD,20);
       Font font2=new Font("TimesRoman",Font.BOLD,15);
       display.setFont(font1);
       display.appendText ("\n      X table       \n");
       display.appendText("   ==========   \n\n");
       display.setFont(font2);
    String[][][] xijk=yz_table.getXijk();
    for(int kk=0;kk<input_k;kk++){
      for(int i=0;i<input_i;i++){
        for(int j=0;j<input_j;j++){
          String s=String.valueOf(xijk[i][j][kk]);
          display.appendText("[ "+s+"]");
        }
        display.appendText("\n");
      }
      display.appendText("\n");
    }
  }
  void button1_actionPerformed(ActionEvent e) {
    System.exit(0);
  }
  JLabel jLabel1 = new JLabel();
  FlowLayout flowLayout1 = new FlowLayout();
  TextField timeEnd = new TextField();
}

public class RjkFrame extends JFrame {
//my variable to goes here
  private MainFrame owner;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  private int input_i,input_j,input_k;
//  private Sik sikFrame;
  BorderLayout borderLayout1 = new BorderLayout();
  Label label1 = new Label();
  Panel panel1 = new Panel();
  GridControl gridControl1 = new GridControl();
  Button okButton = new Button();
  public RjkFrame(MainFrame owner) {
//my code to goes here
    this.owner=owner;
    this.input_i=owner.input_i;
    this.input_j=owner.input_j;
    this.input_k=owner.input_k;
//    this.sikFrame=frame1.sikTable;
    try  {
     jbInit();
    }
    catch(Exception e) {
     e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(360,330));
    this.setTitle("rjk");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    this.setResizable(true);
    label1.setBackground(Color.blue);
    label1.setFont(new java.awt.Font("Dialog", 1, 26));
```

This material is protected by copyright and, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
label1.setForeground(Color.red);
label1.setName("Rjk TABLE");
label1.setAlignment(1);
label1.setText("rjk Table");
this.getContentPane().setLayout(borderLayout1);
panel1.setBackground(Color.blue);
gridControl1.setAutoAppend(true);
gridControl1.setColumnCaptions(new String[] {"1"});
gridControl1.setItems(new String[][] {{""},});
okButton.setFont(new java.awt.Font("Dialog", 1, 16));
okButton.setForeground(Color.red);
okButton.setLabel("  Ok  ");
okButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
    okButton_actionPerformed(e);
  }
});
this.getContentPane().add(label1, BorderLayout.NORTH);
this.getContentPane().add(panel1, BorderLayout.SOUTH);
this.getContentPane().add(gridControl1, BorderLayout.CENTER);
panel1.add(okButton, null);
//---- my code to goes here ------
  addColumns(input_k);
  addRows(input_j);
//Load the stopword file and place individual words in a vector
        BufferedReader in = new BufferedReader(new FileReader("rjk.txt"));
        Vector buf = new Vector(10);
        String s;
    String sx[][] = new String[input_j][input_k];
    int row=0;
    System.out.println("\n===== rjk =====\n");
        while ((s = in.readLine()) != null){
```

This material is protected for education use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    StringTokenizer st = new StringTokenizer(s," ");
    int col = 0;
    while(st.hasMoreTokens()){
      sx[row][col]=st.nextToken();
      System.out.print(sx[row][col]);
      col++;
    }
    System.out.print("\n");
    row++;
   }
          in.close();
   gridControl1.setItems(sx);
 }
 public void addColumns(int col){
  for(int i=2;i<=col;i++){
   gridControl1.addColumn();
   gridControl1.setColumnCaptions(new String[i]);
  }
 }
 public void addRows(int row){
  for(int i=2;i<=row;i++){
   gridControl1.addRow();
//    gridControl1.setRowCaptions(new String[i]);
  }
 }
 void okButton_actionPerformed(ActionEvent e) {
  boolean istrue=true;
  String[][] data=getGridItems();
  int[][] dataInt=new int[input_j][input_k];
  try{
    for(int j=0;j<input_j;j++){
     for(int k=0;k<input_k;k++){
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
        dataInt[j][k]=Integer.parseInt(data[j][k]);
       }
      }
      for(int j=0;j<input_j;j++){
       for(int k=0;k<input_k;k++){
        if(dataInt[j][k]>1||dataInt[j][k]<0){
         istrue=false;
        }
       }
      }
    }catch(NumberFormatException ee){
     istrue=false;
    }
    if(istrue){
     owner.rjkArray=getGridItems();
//------ test ----------
for(int j=0;j<input_j;j++){
  for(int k=0;k<input_k;k++){
   System.out.print("["+owner.rjkArray[j][k]+"]");
  }
  System.out.print("\n");
}
    //set frame to center
     Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
     Dimension frameSize = owner.sikFrame.getSize();
     if (frameSize.height > screenSize.height)
       frameSize.height = screenSize.height;
     if (frameSize.width > screenSize.width)
       frameSize.width = screenSize.width;
     owner.sikFrame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
 frameSize.height) / 2);
     owner.sikFrame.setVisible(true);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
      owner.sikFrame.show();
//    this.hide();
//    this.dispose();
      okButton.disable();
    }
  }
  void this_windowClosing(WindowEvent e) {
    this.hide();
    this.dispose();
  }
  public synchronized String[][] getGridItems(){
    return gridControl1.getItems();
  }
  public void setTextTitle(String s){
    this.setTitle(s);
  }
}

public class SikFrame extends JFrame {
//my variable to goes here
  private boolean isNew=false;
  private MainFrame owner;
  private int input_i,input_j,input_k;
  private int cijkFrameId;
  private Vector vectorCijkFrame;
  private Vector vectorDkFrame;
//  private Vector vk;
//  private Cijk cijkTable;
  BorderLayout borderLayout1 = new BorderLayout();
  Panel panel1 = new Panel();
  Button okButton = new Button();
  Label label1 = new Label();
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
 GridControl gridControl1 = new GridControl();
 public SikFrame(MainFrame owner) {
//---- my code to goes here -----
  this.owner=owner;
  this.input_i=owner.input_i;
  this.input_j=owner.input_j;
  this.input_k=owner.input_k;
  this.cijkFrameId=owner.cijkFrameId;
  this.vectorCijkFrame=owner.vectorCijkFrame;
  this.vectorDkFrame=owner.vectorDkFrame;
//  this.vk=frame1.vk;
   try {
    jbInit();
   }
   catch(Exception e) {
    e.printStackTrace();
   }
  }
  private void jbInit() throws Exception {
   this.setSize(new Dimension(360,330));
   this.setResizable(true);
   this.getContentPane().setLayout(borderLayout1);
   panel1.setBackground(Color.blue);
   panel1.setForeground(Color.blue);
   okButton.setFont(new java.awt.Font("Dialog", 1, 16));
   okButton.setForeground(Color.red);
   okButton.setLabel("  Ok  ");
   okButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
     okButton_actionPerformed(e);
    }
   });
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
label1.setBackground(Color.blue);
label1.setFont(new java.awt.Font("Dialog", 1, 28));
label1.setForeground(Color.red);
label1.setAlignment(1);
label1.setText("sik Table");
this.setTitle("sjk");
this.addWindowListener(new java.awt.event.WindowAdapter() {
 public void windowClosing(WindowEvent e) {
  this_windowClosing(e);
 }
});
this.getContentPane().add(panel1, BorderLayout.SOUTH);
this.getContentPane().add(label1, BorderLayout.NORTH);
this.getContentPane().add(gridControl1, BorderLayout.CENTER);
panel1.add(okButton, null);
//----- my code to goes here -------
addColumns(input_k);
addRows(input_i);
//  setShowButton(false);
//Load the stopword file and place individual words in a vector
            BufferedReader in = new BufferedReader(new FileReader("sik.txt"));
            Vector buf = new Vector(10);
            String s;
    String sx[][] = new String[input_i][input_k];
    int row=0;
    System.out.println("\n====== rjk ======\n");
            while ((s = in.readLine()) != null){
      StringTokenizer st = new StringTokenizer(s," ");
      int col = 0;
      while(st.hasMoreTokens()){
       sx[row][col]=st.nextToken();
       System.out.print(sx[row][col]);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
      col++;
     }
     System.out.print("\n");
     row++;
    }
            in.close();
    gridControl1.setItems(sx);
}
public void addColumns(int col){
  for(int i=2;i<=col;i++){
     gridControl1.addColumn();
  }
}
 public void addRows(int row){
   for(int i=2;i<=row;i++){
      gridControl1.addRow();
   }
 }
 void preButton_actionPerformed(ActionEvent e) {
 }
 void okButton_actionPerformed(ActionEvent e) {
 if(!isNew){
    StandardDialog1 standardDialog1=new StandardDialog1(this,"Dialog box",true,owner);
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = standardDialog1.getSize();
   owner.dup=true;
//--- after user prompt option K table ---
//--- create CijkFrame ----------
    int n=0;
    owner.cijkFrameId=0; //initialize for zero
    CijkFrame cijkFrame;
    while(n<input_k){
```

This content is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    cijkFrame=new CijkFrame(owner);
    vectorCijkFrame.addElement(cijkFrame);
    n++;
    owner.cijkFrameId++;
  }
  CijkFrame  cijk=(CijkFrame)vectorCijkFrame.elementAt(0);
  screenSize = Toolkit.getDefaultToolkit().getScreenSize();
  frameSize = cijk.getSize();
  if (frameSize.height > screenSize.height)
    frameSize.height = screenSize.height;
  if (frameSize.width > screenSize.width)
    frameSize.width = screenSize.width;
  cijk.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
  cijk.setVisible(true);
  cijk.show();
  owner.sikArray=getGridItems();
//------ test ------
for(int i=0;i<input_i;i++){
 for(int k=0;k<input_k;k++){
  System.out.print("["+owner.sikArray[i][k]+"]");
 }
 System.out.print("\n");
}
  okButton.disable();
 }else{
  CijkFrame c=(CijkFrame)vectorCijkFrame.elementAt(0);
  c.show_new();
  c.setPrevButton(false);
  c.setOkButton(false);
  c.setNextButton(true);
  c.setNew(true);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  c.show();
  owner.sikArray=getGridItems();
 }
}
 void this_windowClosing(WindowEvent e) {
  this.hide();
  this.dispose();
 }
 public void fillGridItems(){
   int[] dkArray=new int[input_k+8];
   int count=0;
   Enumeration enum=vectorDkFrame.elements();
   DkFrame dk;
   while(enum.hasMoreElements()){
    dk=(DkFrame)enum.nextElement();
    int[] tmp=dk.getItems();
    for(int i=0;i<8;i++){
      //if(i+count<input_k)dkArray[i+count]=tmp[i];
      dkArray[i+count]=tmp[i];
    }
    count+=8;
   }
   String[][] items=new String[input_i][input_k];
   for(int r=0;r<input_i;r++){
    for(int c=0;c<input_k;c++){
      items[r][c]=String.valueOf(Integer.parseInt(owner.sikArray[r][c])*dkArray[c]);
      gridControl1.setItems(items);
    }
   }
  }
  public void show_new() {
   owner.sikArray=getGridItems();
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    fillGridItems();
//    Show.disable();
//    nextButton.enable();
  }
  public synchronized String[][] getGridItems(){
      return gridControl1.getItems();
  }
  public void setOkButton(boolean b){
    if(b)okButton.enable();
    else okButton.disable();
  }
  public void setNew(boolean n){
    isNew=n;
  }
}

public class StandardDialog1 extends JDialog {
//-- my variable to goes here ---
  MainFrame owner;
  boolean dup;
  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();
  JButton yesButton = new JButton();
  JButton noButton = new JButton();
  Border border1;
  JPanel jPanel1 = new JPanel();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  GridLayout gridLayout1 = new GridLayout();
  Label label1 = new Label();
  public StandardDialog1(Frame frame, String title, boolean modal,MainFrame owner) {
    super(frame, title, modal);
    this.owner=owner;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  this.dup=owner.dup;
  try {
   jbInit();
  }
  catch (Exception e) {
    e.printStackTrace();
  }
  pack();
}
private void jbInit() throws Exception {
  this.setSize(new Dimension(360,330));
  border1 = BorderFactory.createRaisedBevelBorder();
  jPanel1.setLayout(gridLayout1);
  panel2.setBorder(border1);
  panel2.setLayout(null);
  yesButton.setText("Yes");
  yesButton.addActionListener(new StandardDialog1_yesButton_actionAdapter(this));
  noButton.setText("No");
  gridLayout1.setHgap(4);
  noButton.addActionListener(new StandardDialog1_noButton_actionAdapter(this));
  this.addWindowListener(new StandardDialog1_this_windowAdapter(this));
  panel1.setLayout(gridBagLayout1);
  label1.setBounds(new Rectangle(34, 34, 252, 37));
  label1.setFont(new java.awt.Font("Dialog", 1, 18));
  label1.setForeground(Color.red);
  label1.setAlignment(1);
  label1.setText("The table of  K  is duplicate");
  panel1.add(jPanel1, new GridBagConstraints(0, 1, 1, 1, 1.0, 0.0
        ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(4, 8, 4, 8), 0, 0));
  jPanel1.add(yesButton, null);
  jPanel1.add(noButton, null);
  panel1.add(panel2, new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
        ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(2, 18, 0, 2), 319,
114));
   panel2.add(label1, null);
   getContentPane().add(panel1);
}
 // OK
 void yesButton_actionPerformed(ActionEvent e) {
   owner.dup=true;
if(owner.dup==true) System.out.println("dup=true\n");
else System.out.println("dup!=true\n");
   dispose();
 }
 // Cancel
 void noButton_actionPerformed(ActionEvent e) {
   owner.dup=false;
if(owner.dup==false)System.out.println("dup=false\n");
else System.out.println("dup!=false\n");
   dispose();
 }
 void this_windowClosing(WindowEvent e) {
   dispose();
 }
}
class StandardDialog1_yesButton_actionAdapter implements ActionListener {
 StandardDialog1 adaptee;
 StandardDialog1_yesButton_actionAdapter(StandardDialog1 adaptee) {
   this.adaptee = adaptee;
 }
 public void actionPerformed(ActionEvent e) {
   adaptee.yesButton_actionPerformed(e);
 }
}
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
class StandardDialog1_noButton_actionAdapter implements ActionListener {
 StandardDialog1 adaptee;
 StandardDialog1_noButton_actionAdapter(StandardDialog1 adaptee) {
  this.adaptee = adaptee;
 }
 public void actionPerformed(ActionEvent e) {
  adaptee.noButton_actionPerformed(e);
 }
}
class StandardDialog1_this_windowAdapter extends WindowAdapter {
 StandardDialog1 adaptee;
 StandardDialog1_this_windowAdapter(StandardDialog1 adaptee) {
  this.adaptee = adaptee;
 }
 public void windowClosing(WindowEvent e) {
  adaptee.this_windowClosing(e);
 }
}

public class TFrame extends JFrame {
//--- my variable to goes here ---
  private MainFrame owner;
  private Vector vectorTFrame;
  private Vector vField=new Vector();
  private int input_i,input_j,input_k;
  public  int[] N=new int[8];
  private RjkFrame rjkFrame;
  public  int frameId=0;
  Button prevButton = new Button();
  Button okButton = new Button();
  Button nextButton = new Button();
  Label label1 = new Label();
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
Label n1Label = new Label();
Label n2Label = new Label();
Label n3Label = new Label();
Label n4Label = new Label();
Label n5Label = new Label();
Label n6Label = new Label();
Label n7Label = new Label();
Label n8Label = new Label();
TextField n1Field = new TextField();
TextField n2Field = new TextField();
TextField n3Field = new TextField();
TextField n4Field = new TextField();
TextField n5Field = new TextField();
TextField n6Field = new TextField();
TextField n7Field = new TextField();
TextField n8Field = new TextField();
public TFrame(MainFrame owner) {
//--------- my code to goes here ----------
  this.owner=owner;
  this.rjkFrame=owner.rjkFrame;
  this.vectorTFrame=owner.vectorTFrame;
  this.input_i=owner.input_i;
  this.input_j=owner.input_j;
  this.input_k=owner.input_k;
  this.frameId=owner.tFrameId;
  try  {
   jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
}
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
private void jbInit() throws Exception {
 this.setSize(new Dimension(360,330));
 this.setTitle("Ni Table");
 this.addWindowListener(new java.awt.event.WindowAdapter() {
  public void windowClosing(WindowEvent e) {
   this_windowClosing(e);
  }
 });
 prevButton.setBackground(Color.lightGray);
 prevButton.setBounds(new Rectangle(81, 232, 67, 30));
 prevButton.setFont(new java.awt.Font("Dialog", 1, 12));
 prevButton.setForeground(Color.black);
 prevButton.setLabel("<< Prev");
 prevButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
   prevButton_actionPerformed(e);
  }
 });
 this.getContentPane().setLayout(null);
 okButton.setBackground(Color.lightGray);
 okButton.setBounds(new Rectangle(152, 232, 44, 30));
 okButton.setFont(new java.awt.Font("Dialog", 1, 14));
 okButton.setForeground(Color.black);
 okButton.setLabel("Ok");
 okButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
   okButton_actionPerformed(e);
  }
 });
 nextButton.setBackground(Color.lightGray);
 nextButton.setBounds(new Rectangle(200, 232, 66, 30));
 nextButton.setFont(new java.awt.Font("Dialog", 1, 12));
```

This thesis is not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
nextButton.setForeground(Color.black);
nextButton.setLabel("Next >>");
nextButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  nextButton_actionPerformed(e);
 }
});
label1.setBounds(new Rectangle(89, 12, 137, 29));
label1.setFont(new java.awt.Font("Dialog", 1, 28));
label1.setForeground(Color.red);
label1.setAlignment(1);
label1.setText("Input  Ni");
n1Label.setBounds(new Rectangle(30, 70, 24, 25));
n1Label.setFont(new java.awt.Font("Dialog", 1, 16));
n1Label.setForeground(Color.red);
n1Label.setText("N 1");
n2Label.setBounds(new Rectangle(30, 100, 24, 26));
n2Label.setFont(new java.awt.Font("Dialog", 1, 16));
n2Label.setForeground(Color.red);
n2Label.setText("N 2");
n3Label.setBounds(new Rectangle(30, 130, 23, 27));
n3Label.setFont(new java.awt.Font("Dialog", 1, 16));
n3Label.setForeground(Color.red);
n3Label.setText("N 3");
n4Label.setBounds(new Rectangle(30, 160, 25, 26));
n4Label.setFont(new java.awt.Font("Dialog", 1, 16));
n4Label.setForeground(Color.red);
n4Label.setText("N 4");
n5Label.setBounds(new Rectangle(170, 70, 24, 26));
n5Label.setFont(new java.awt.Font("Dialog", 1, 16));
n5Label.setForeground(Color.red);
n5Label.setText("N5");
```

This is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
n6Label.setBounds(new Rectangle(170, 102, 25, 20));
n6Label.setFont(new java.awt.Font("Dialog", 1, 16));
n6Label.setForeground(Color.red);
n6Label.setText("N6");
n7Label.setBounds(new Rectangle(170, 132, 24, 23));
n7Label.setFont(new java.awt.Font("Dialog", 1, 16));
n7Label.setForeground(Color.red);
n7Label.setText("N7");
n8Label.setBounds(new Rectangle(169, 167, 24, 21));
n8Label.setFont(new java.awt.Font("Dialog", 1, 16));
n8Label.setForeground(Color.red);
n8Label.setText("N8");
n1Field.setBounds(new Rectangle(60, 72, 71, 18));
n1Field.setEditable(false);
n1Field.setText("4");
n2Field.setBounds(new Rectangle(59, 103, 71, 19));
n2Field.setEditable(false);
n2Field.setText("4");
n3Field.setBounds(new Rectangle(59, 135, 72, 19));
n3Field.setEditable(false);
n4Field.setBounds(new Rectangle(58, 166, 73, 19));
n4Field.setEditable(false);
n5Field.setBounds(new Rectangle(201, 71, 73, 20));
n5Field.setEditable(false);
n6Field.setBounds(new Rectangle(201, 102, 74, 20));
n6Field.setEditable(false);
n7Field.setBounds(new Rectangle(200, 135, 73, 20));
n7Field.setEditable(false);
n8Field.setBounds(new Rectangle(199, 167, 73, 21));
n8Field.setEditable(false);
this.getContentPane().add(label1, null);
this.getContentPane().add(n1Label, null);
```

This material is protected by copyright. Educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    this.getContentPane().add(n2Label, null);
    this.getContentPane().add(n3Label, null);
    this.getContentPane().add(n4Label, null);
    this.getContentPane().add(n5Label, null);
    this.getContentPane().add(n7Label, null);
    this.getContentPane().add(n8Label, null);
    this.getContentPane().add(n1Field, null);
    this.getContentPane().add(n2Field, null);
    this.getContentPane().add(n3Field, null);
    this.getContentPane().add(n4Field, null);
    this.getContentPane().add(n5Field, null);
    this.getContentPane().add(n6Field, null);
    this.getContentPane().add(n7Field, null);
    this.getContentPane().add(n8Field, null);
    this.getContentPane().add(okButton, null);
    this.getContentPane().add(nextButton, null);
    this.getContentPane().add(prevButton, null);
    this.getContentPane().add(n6Label, null);
//---- my code to goes here ----
    vField.addElement(n1Field);
    vField.addElement(n2Field);
    vField.addElement(n3Field);
    vField.addElement(n4Field);
    vField.addElement(n5Field);
    vField.addElement(n6Field);
    vField.addElement(n7Field);
    vField.addElement(n8Field);
    n1Label.setText("N"+(1+(getFrameId()*8)));
    n2Label.setText("N"+(2+(getFrameId()*8)));
    n3Label.setText("N"+(3+(getFrameId()*8)));
    n4Label.setText("N"+(4+(getFrameId()*8)));
    n5Label.setText("N"+(5+(getFrameId()*8)));
```

This material is for educational purpose only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    n6Label.setText("N"+(6+(getFrameId()*8)));
    n7Label.setText("N"+(7+(getFrameId()*8)));
    n8Label.setText("N"+(8+(getFrameId()*8)));
    int i=getFrameId()*8;
    int size=0;
    TextField t;
//System.out.println("enter while loop -> i="+i+" -> input_i="+input_i+"\n");
    while((i<input_i/*(input_i-(getFrameId()*8))*/)&&(size<vField.size())){
      t=(TextField)vField.elementAt(size);
      t.setEditable(true);
//System.out.println(t.getText()+i+" t\n");
      i++;
      size++;
    }
 System.out.println("exit while loop\n");
    if(0==getFrameId()){
      prevButton.disable();
      nextButton.enable();
    }
    if(input_i-getFrameId()*8<=8){
      nextButton.disable();
//      prevButton.enable();
    }
    for(int ii=0;ii<8;ii++){
      N[ii]=0;
    }
  }
  public void setFrameId(int id){
    frameId=id;
  }
  public int getFrameId(){
    return frameId;
```

This content is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
}
void nextButton_actionPerformed(ActionEvent e) {
  try{
     this.N[0]=Integer.parseInt(n1Field.getText());
     this.N[1]=Integer.parseInt(n2Field.getText());
     this.N[2]=Integer.parseInt(n3Field.getText());
     this.N[3]=Integer.parseInt(n4Field.getText());
     this.N[4]=Integer.parseInt(n5Field.getText());
     this.N[5]=Integer.parseInt(n6Field.getText());
     this.N[6]=Integer.parseInt(n7Field.getText());
     this.N[7]=Integer.parseInt(n8Field.getText());
   }catch(NumberFormatException ee){}
   TFrame tFrame=(TFrame)vectorTFrame.elementAt(getFrameId()+1);  //next element of
TFrmae
//System.out.println("frameId="+getFrameId());
   Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
   Dimension frameSize = tFrame/*frame*/.getSize();
   if (frameSize.height > screenSize.height)
     frameSize.height = screenSize.height;
   if (frameSize.width > screenSize.width)
     frameSize.width = screenSize.width;
   tFrame/*frame*/.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
   tFrame/*frame*/.setVisible(true);
   tFrame.show();
   this.hide();
//    this.dispose();
  }
  void okButton_actionPerformed(ActionEvent e) {
    try{
      if(!n1Field.getText().equals(""))
        this.N[0]=Integer.parseInt(n1Field.getText());
```

This is for educational purpose only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
else this.N[0]=0;
if(!n2Field.getText().equals(""))
 this.N[1]=Integer.parseInt(n2Field.getText());
else this.N[1]=0;
if(!n3Field.getText().equals(""))
 this.N[2]=Integer.parseInt(n3Field.getText());
else this.N[2]=0;
if(!n4Field.getText().equals(""))
 this.N[3]=Integer.parseInt(n4Field.getText());
else this.N[3]=0;
if(!n5Field.getText().equals(""))
 this.N[4]=Integer.parseInt(n5Field.getText());
else this.N[4]=0;
if(!n6Field.getText().equals(""))
 this.N[5]=Integer.parseInt(n6Field.getText());
else this.N[5]=0;
if(!n7Field.getText().equals(""))
 this.N[6]=Integer.parseInt(n7Field.getText());
else this.N[6]=0;
if(!n8Field.getText().equals(""))
 this.N[7]=Integer.parseInt(n8Field.getText());
else this.N[7]=0;
int x = this.N[0]+this.N[1]+this.N[2]+this.N[3]+this.N[4]+this.N[5]+this.N[6]+this.N[7];
System.out.println("x:="+x+"  j:="+input_j);
if( x < input_j){
    n1Field.setText("");
    n2Field.setText("");
    n3Field.setText("");
    n4Field.setText("");
    n5Field.setText("");
    n6Field.setText("");
    n7Field.setText("");
```

This material is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
        n8Field.setText("");
        this.N[0]=0;
        this.N[1]=0;
        this.N[2]=0;
        this.N[3]=0;
        this.N[4]=0;
        this.N[5]=0;
        this.N[6]=0;
        this.N[7]=0;
        return;
      }
    }catch(NumberFormatException ee){
     System.out.println(ee.toString());
    }
//----- set rjkFrame alignment to center -------
//    this.hide();
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = owner.rjkFrame.getSize();//frame.getSize();
    if (frameSize.height > screenSize.height)
      frameSize.height = screenSize.height;
    if (frameSize.width > screenSize.width)
      frameSize.width = screenSize.width;
    owner.rjkFrame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
 frameSize.height) / 2);
    owner.rjkFrame.setVisible(true);
    owner.rjkFrame.show();
//    this.hide();
//    this.dispose();
    okButton.disable();
   }
   void prevButton_actionPerformed(ActionEvent e) {
     TFrame tFrame=(TFrame)vectorTFrame.elementAt(getFrameId()-1);
```

This material is protected by Rajamangala University of Technology Ladkrabang for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    tFrame.show();
    this.hide();
//    this.dispose();
  }
  public int getNi(int i){
    if(i<0)
      return -2;
    else if(i<8){
      return N[i];
    }else
      return -1;
  }
  void this_windowClosing(WindowEvent e) {
    this.hide();
    this.dispose();
  }
}

public class VjkFrame extends JFrame {
  Label label1 = new Label();
  GridControl gridControl1 = new GridControl();
  Button okButton_vjkFrame = new Button();
//----- my variable -----
  private int input_j,input_k;
  private MainFrame owner;
  private Vector vectorDkFrame;
  private NjkFrame njkFrame;
  private Vector vectorCijkFrame;
  public VjkFrame(MainFrame owner) {
    this.owner=owner;
    this.vectorDkFrame=owner.vectorDkFrame;
    this.njkFrame=owner.njkFrame;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    this.input_j=owner.input_j;
    this.input_k=owner.input_k;
    this.vectorCijkFrame=owner.vectorCijkFrame;
    try {
      jbInit();
//    readGridControl();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(360,330));
    this.setFont(new java.awt.Font("Dialog", 1, 20));
    this.setResizable(true);
    this.setTitle("vjk");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    this.setResizable(true);
    label1.setBounds(new Rectangle(103, 8, 188, 31));
    label1.setFont(new java.awt.Font("Dialog", 1, 28));
    label1.setForeground(Color.red);
    label1.setAlignment(1);
    label1.setText("vjk Table");
    this.getContentPane().setLayout(null);
    this.setFont(new java.awt.Font("Dialog", 0, 20));
    okButton_vjkFrame.setBounds(new Rectangle(151, 257, 68, 32));
    okButton_vjkFrame.setFont(new java.awt.Font("Dialog", 1, 16));
    okButton_vjkFrame.setForeground(Color.red);
```

This material is protected by copyright and is for educational purposes only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  okButton_vjkFrame.setLabel("Ok");
  okButton_vjkFrame.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      okButton_vjkFrame_actionPerformed(e);
    }
  });
  gridControl1.setBounds(new Rectangle(25, 44, 313, 198));
  this.getContentPane().add(label1, null);
  this.getContentPane().add(gridControl1, null);
  this.getContentPane().add(okButton_vjkFrame, null);
//---------- my program ----------
   addColumns(input_k);
   addRows(input_j);
//   String[][] njkItems=njkFrame.gridControl1.getItems());
 }
 public void addColumns(int col){
   for(int i=2;i<=col;i++){
     gridControl1.addColumn();
     gridControl1.setColumnCaptions(new String[i]);
   }
 }
 public void addRows(int row){
   for(int i=2;i<=row;i++){
     gridControl1.addRow();
//     gridControl1.setRowCaptions(new String[i]);
   }
 }
 void this_windowClosing(WindowEvent e) {
   this.hide();
   this.dispose();
 }
 void okButton_vjkFrame_actionPerformed(ActionEvent e) {
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    owner.vjkArray=gridControl1.getItems();
    //----- set all object for enable Show button -----
    Enumeration enum=owner.vectorCijkFrame.elements();
    while(enum.hasMoreElements()){
      CijkFrame cijk=(CijkFrame)enum.nextElement();
//      cijk.setShowButton(true);
    }
    owner.sikFrame.setNew(true);
    owner.sikFrame.setOkButton(true);
    owner.sikFrame.show_new();
    owner.sikFrame.show();
  }//end of ok_button action
  public void fillGridItems(){
    int[] dkArray=new int[input_k];
    int count=0;
    Enumeration enum=vectorDkFrame.elements();
    DkFrame dk;
    while(enum.hasMoreElements()){
      dk=(DkFrame)enum.nextElement();
      int[] tmp=dk.getItems();
      for(int i=0;i<8;i++){
        if(i+count<input_k)dkArray[i+count]=tmp[i];
      }
      count+=8;
    }
    String[][] items=new String[input_j][input_k];
    for(int r=0;r<input_j;r++){
      for(int c=0;c<input_k;c++){
        //System.out.println("njkArray["+r+"]["+c+"]="+owner.njkArray[r][c]);
        items[r][c]=String.valueOf(Integer.parseInt(owner.njkArray[r][c])*dkArray[c]);
        gridControl1.setItems(items);
      }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    }
   }
}

public class YZ_table {
  public int[][] yArray,knArray,tmpArray,maxArray,zArray,minZarray;
  public int input_i,input_j,input_k;
  private MainFrame owner;
  private Vector vectorTFrame;
  private int yArrayMaxCol;
  private int[] NiArray;
  private int[][] valueA,valueB,minValueB,minValueA;
  private int[] sumOfA_B,minSumOfA_B,totalMinSumOfA_B;
  public Vector vectorCijkGridItems;
  private boolean isFirstCallTestNset=true;
  private int totalMin;
  private Date theDate;
  private long  mm1,mm2,ss1,ss2,hh1,hh2;
  Report report;
  //Disp disp=new Disp("zArray");
  //Disp t=new Disp("yArray");
  //Disp tmp=new Disp("tmpArray");
  //Disp v=new Disp("valid");
  private Disp ydisp;
  //private Disp indexdisp=new Disp("index");
  private int count;
  private CijkFrame cijk;
  private CijkFrame[] minCijk;
  private Enumeration enum;
  private String[][] c,tmpC;
  private String[][][] minC,totalMinC;
  public String stime,etime;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
public YZ_table(MainFrame owner) {
  //ydisp=new Disp("yArray");
  //indexdisp=new Disp("index");
  count=0;
   this.owner=owner;
   this.input_i=owner.input_i;
   this.input_j=owner.input_j;
   this.input_k=owner.input_k;
   this.vectorTFrame=owner.vectorTFrame;
   NiArray=new int[input_i];
   fillNi(); //fill array by input_Ni of TFrame
   tmpArray=new int[input_i][input_j];
   tmpC=new String[input_i][input_j];
   totalMinC=new String[input_i][input_j][input_k];
   minC=new String[input_i][input_j][input_k];
   zArray=new int[input_i][input_j];
   minZarray=new int[input_i][input_j];
   valueB=new int[input_i][input_k];
   minValueB=new int[input_i][input_k];
   valueA=new int[input_j][input_k];
   minValueA=new int[input_j][input_k];
   sumOfA_B=new int[input_k];
   minSumOfA_B=new int[input_k];
//    totalMinSumOfA_B=new int[input_k];
   vectorCijkGridItems=new Vector();
   yArrayMaxCol=power(2,input_i); //maxCol=2^i
   yArray=new int[input_i+1][yArrayMaxCol]; // lastest row indicate valid column
   minCijk=new CijkFrame[input_k];
   cijk = new CijkFrame(owner);
//    minC=new String[input_i][input_j][input_k];
   fillYarray(input_i,yArrayMaxCol);
  System.out.println("\n ..... yArray.....");
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
for(int tr=0;tr<input_i;tr++){
  for(int tc=0;tc<yArrayMaxCol;tc++){
    System.out.print("["+yArray[tr][tc]+"]");
  }
  System.out.print("\n");
}
   totalMin=0; //initial state.
   hh1=getGMThours();
   mm1=getGMTminutes();//Integer.parseInt(smin);
   ss1=getGMTseconds();//Integer.parseInt(ssec);
   theDate=new Date();
   stime=theDate.toGMTString();
   fillZarray();
   hh2=getGMThours();
   mm2=getGMTminutes();//Integer.parseInt(smin2);
   ss2=getGMTseconds();//Integer.parseInt(ssec2);
   System.out.println("total time : "+getTotalTime()+" seconds.");
   theDate=new Date();
   etime=theDate.toGMTString();
   report=new Report(this);
   //****** Print to file result.txt ******************
   try{
   File file = new File("result.txt");
   file.delete();
   PrintStream out = new PrintStream(new AppendFileStream("result.txt"));
   out.println("result.txt");
   out.print("\nTotal Cost    : "+getCost());
   out.print("\nStarting  Time\t: "+stime);
   out.print("\nTerminate Time\t: "+etime+"\n");
   int[][] minz=getMinZarray();
       out.print("\n      Z table       \n");
       out.print("      ============      \n\n");
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
for(int i=0;i<input_i;i++){
  for(int j=0;j<input_j;j++){
    int value=minz[i][j];
    String s=String.valueOf(value);
    out.print(" [ "+s+"]");
  }//j
  out.print("\n");
}//i
    out.print("\n      Y table       \n");
    out.print("    ==========    \n\n");
 String[][] yin=getYin();
 for(int i=0;i<input_i;i++){
  for(int kk=0;kk<input_k;kk++){
    String s=String.valueOf(yin[i][kk]);
    out.print("[ "+s+"]");
  }
  out.print("\n");
 }
    out.print("\n      X table       \n");
    out.print("    ==========    \n");
 String[][][] xijk=getXijk();
 for(int kk=0;kk<input_k;kk++){
  out.print("\nk:= "+(kk+1)+"\n");
  for(int i=0;i<input_i;i++){
    for(int j=0;j<input_j;j++){
      String s=String.valueOf(xijk[i][j][kk]);
      out.print("[ "+s+"]");
    }
    out.print("\n");
  }
  out.print("\n");
 }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    out.print("\n");
    out.close();
   }catch(Exception e){
   }
//    testNset();
//     zArray=z2;
//     testNset();
  }
  public int getCost(){return totalMin;}
  public long getTotalTime(){
   return ( ((hh2*60*60)+(mm2*60)+ss2)-((hh1*60*60)+(mm1*60)+ss1) );
  }
 public int getGMThours(){
  theDate=new Date();
  return theDate.getHours();
 }
 public int getGMTminutes(){
                    theDate=new Date();
          //return theDate.toGMTString().substring(15,17);
    //System.out.println("Day: "+theDate.toGMTString());
                    //return theDate.toGMTString().substring(14,16);
    return theDate.getMinutes();
          }
          public int getGMTseconds(){
                    theDate=new Date();
                    //return theDate.toGMTString().substring(18,20);
    //System.out.println("Day: "+theDate.toGMTString());
                    //return theDate.toGMTString().substring(17,19);
    return theDate.getSeconds();
          }
   public void testNset(){
    boolean[] isFirstCallSumOfA_B=new boolean[input_k];//{true,true,true};
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
boolean isFirst=true;
boolean valid=true;
int row=0,col=0;
//try{
 //PrintStream out = new PrintStream(new AppendFileStream("result.txt"));
 //out.println("result.txt");
for(int ii=0;ii<input_k;ii++){
  isFirstCallSumOfA_B[ii]=true;
}
for(int y=0;y<yArrayMaxCol;y++){
   for(int j=0;j<input_j;j++){
      for(int i=0;i<input_i;i++){
        tmpArray[i][j]=yArray[i][y]*zArray[i][j];
      }
   }
  valid=true;
  col=0;
   while(valid&col<input_j){
      valid=false;
      row=0;
      while(!valid&&row<input_i){ // once column
        if(tmpArray[row][col]==1){
         valid=true;
        }
        row++;
      }
      col++;
    }
    if(valid){
      yArray[input_i][y]=1; // valid column/
      //out.print("yArray["+input_i+"]["+y+"] is valid");
    }else yArray[input_i][y]=0;//invalid column.
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    }
    enum=owner.vectorCijkFrame.elements();
    int k=0;
    while(enum.hasMoreElements()){
     //System.out.println("wwwwwww");
     cijk=(CijkFrame)enum.nextElement();
//     if(isFirstCallSumOfA_B[k])minCijk[k]=getMinCijk(cijk,cijk.getGridItems());
     for(int n=0;n<yArrayMaxCol;n++){
      c=cijk.getGridItems();
      for(int i=0;i<input_i;i++){
       for(int j=0;j<input_j;j++){
        tmpC[i][j]=c[i][j];
       }
      }
      if(yArray[input_i][n]==1 || checkForRjk(n,k)){ // value "1" is valid column.
       // if(checkForRjk(n,k))out.println("==========valid...==========");
       for(int i=0;i<input_i;i++){
        for(int j=0;j<input_j;j++){
         try{
          tmpC[i][j]=String.valueOf(zArray[i][j]*Integer.parseInt(owner.rjkArray[j][k])*
                    Integer.parseInt(tmpC[i][j])*yArray[i][n]);
          // tmp.setMessage("{"+tmpC[i][j]+"}");
          //out.println("{"+tmpC[i][j]+"}");
         }catch(NumberFormatException ee){
          System.err.println("error:"+ee.toString());
          //tmp.setMessage(ee.toString());
         }
        }//for j
        //tmp.setMessage("\n");
        //out.println();
       }//for i
       if(isFirstCallSumOfA_B[k]){
```

This r[illegible]cational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
sumOfA_B[k]=getSumOfA_B(tmpC,yArray,n,k);
isFirstCallSumOfA_B[k]=false;
//tmp.setMessage("========show First MinC==========");
//System.out.println("========show First MinC==========");
for(int rows=0;rows<input_i;rows++){
 for(int cols=0;cols<input_j;cols++){
  minC[rows][cols][k]=tmpC[rows][cols];
  //tmp.setMessage("["+minC[rows][cols]+"]");
  //System.out.print("["+minC[rows][cols]+"]");
 }
 //tmp.setMessage("\n");
 //System.out.println("");
}
int[][] tmpvb=getValueB(yArray,n,k);
for(int rows=0;rows<input_i;rows++){
 valueB[rows][k]=tmpvb[rows][k];
}
int[][] tmpva=getValueA(tmpC,k);
for(int cols=0;cols<input_j;cols++){
 valueA[cols][k]=tmpva[cols][k];
}
}else if(sumOfA_B[k]>getSumOfA_B(tmpC,yArray,n,k)){ //not first.
  sumOfA_B[k]=getSumOfA_B(tmpC,yArray,n,k);
  //System.out.println("========show update MinC==========");
  for(int rows=0;rows<input_i;rows++){
    for(int cols=0;cols<input_j;cols++){
      minC[rows][cols][k]=tmpC[rows][cols];
      //System.out.print("["+rows+"]["+cols+"]");
    }
    //System.out.print("\n");
  }
  int[][] tmpvb=getValueB(yArray,n,k);
```

This material is provided for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
        //*indexdisp.setMessage("\nk = "+k+"  sumofA_B = "+sumOfA_B[k]);
        for(int rows=0;rows<input_i;rows++){
          valueB[rows][k]=tmpvb[rows][k];
          //*indexdisp.setMessage(" ["+valueB[rows][k]+"]");
        }
       }//end if, not first.
      }//if valid col.
     }//for n
     k++;
    }//while, all cijk.
    if(isFirstCallTestNset){
     totalMin=getTotalMin(sumOfA_B);
     isFirstCallTestNset=false;
//     minSumOfA_B=sumOfA_B;
    }
    if((totalMin==0)||(totalMin>getTotalMin(sumOfA_B))){
     totalMin=getTotalMin(sumOfA_B);
     for(int kk=0;kk<input_k;kk++){
      minSumOfA_B[kk]=sumOfA_B[kk];
     }
     //indexdisp.setMessage("enter set minValueB\n");
     for(int kk=0;kk<input_k;kk++){
      for(int rows=0;rows<input_i;rows++){
        minValueB[rows][kk]=valueB[rows][kk];
        //indexdisp.setMessage("["+minValueB[rows][kk]+"]");
      }
      //indexdisp.setMessage("\n");
      for(int cols=0;cols<input_j;cols++){
       minValueA[cols][kk]=valueA[cols][kk];
      }
     }//for kk
     //System.out.println("-------- totalMinC ----------");
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    for(int kk=0;kk<input_k;kk++){
     for(int i=0;i<input_i;i++){
      for(int j=0;j<input_j;j++){
       totalMinC[i][j][kk]=minC[i][j][kk];
      // System.out.print("["+totalMinC[i][j][kk]+"]");
       //ydisp.setMessage("["+totalMinC[i][j][kk]+"]");
      }
      //System.out.println("");
      //ydisp.setMessage("\n");
     }
     //System.out.println("totalMinC");
     //ydisp.setMessage("\n");
    }
    for(int i=0;i<input_i;i++){
     for(int j=0;j<input_j;j++){
      minZarray[i][j]=zArray[i][j];
     }
    }
   }//end if.
//   for(int kk=0;kk<input_k;kk++){
//    ydisp.setMessage("k="+kk+"sumOfA_B: "+sumOfA_B[kk]+"  :minSumOfA_B:
"+minSumOfA_B[kk]+"\n");
//   }
//   ydisp.setMessage("sumOfA_B="+getTotalMin(sumOfA_B)+"  :totalMin="+totalMin+"\n");
   //out.close();
   //}catch(IOException ex){
    //System.out.println(ex.toString());
   //}
  }
  private boolean checkForRjk(int n, int k){
   int[][] tmpZ=new int[input_i][input_j];
   for(int j=0;j<input_j;j++){
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  for(int i=0;i<input_i;i++){
    tmpZ[i][j]=yArray[i][n]*zArray[i][j];
  }
 }
 boolean valid=true;
 int j=0;
 while(valid & j<input_j){
  int i=0;
  valid = false;
  while(!valid && i<input_i){
   if(tmpZ[i][j] == 0)valid = false;
   else valid = true;
   i++;
  }
  if(!valid && Integer.parseInt(owner.rjkArray[j][k])==1)
   valid=false;
  else valid = true;
  j++;
 }
 return valid;
}
private String[][][] checkMinCol(String[][][] minc){
 int current=0;
 for(int kk=0;kk<input_k;kk++){
 for(int j=0;j<input_j;j++){
  current=0;
  for(int i=1;i<input_i;i++){
   // System.out.println("first="+minc[current][j][kk]);
    int first=Integer.parseInt(minc[current][j][kk]);
    int second=Integer.parseInt(minc[i][j][kk]);
    if(first==0){
     current=i;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    }else if((second!=0)&&(first>second)){
      minc[current][j][kk]="0";
      current=i;
    }else{
      minc[i][j][kk]="0";
    }
   }//for i;
   if(Integer.parseInt(minc[current][j][kk])>0)minc[current][j][kk]="1";
  }//for j;
  }//for kk;
  return minc;
}
private String[][] checkMinRow(int[][] minvb){
  String[][] yn=new String[input_i][input_k];
  //indexdisp.setMessage("\nenter checkMinRow\n");
  for(int i=0;i<input_i;i++){
    for(int kk=0;kk<input_k;kk++){
      //indexdisp.setMessage("["+minvb[i][kk]+"]");
      if(minvb[i][kk]>0){
        yn[i][kk]="1";
      }else{
        yn[i][kk]="0";
      }//if;
    }
    //indexdisp.setMessage("\n");
  }
  return yn;
 }
 private CijkFrame getMinCijk(CijkFrame cijkf,String[][] ca){
  cijkf.gridControl1.setItems(ca);
  return cijkf;
 }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
private int getTotalMin(int[] sum){
  int total=0;
  for(int k=0;k<input_k;k++){
    total=total+sum[k];
  }
  return total;
}
private int[][] getValueA(String[][] cc,int k){
  int A[][]=new int[input_j][input_k];
  int min=0,maybe=0;
  for(int j=0;j<input_j;j++){
    min=Integer.parseInt(cc[0][j]);
    for(int i=1;i<input_i;i++){
      maybe=Integer.parseInt(cc[i][j]);
      if(((maybe!=0)&&(maybe<min))||(min==0)){
        min=maybe;
      }
    }
    A[j][k]=min;
  }
  return A;
}
private int[][] getValueB(int[][] y,int n,int k){
  int[][] B=new int[input_i][input_k];
  for(int i=0;i<input_i;i++){
    B[i][k]=y[i][n]*Integer.parseInt(owner.sikArray[i][k]);
  }
  return B;
}
private int getSumOfA_B(String[][] cc,int[][] y,int n,int k){
    int[][] A=getValueA(cc,k);
    int[][] B=getValueB(y,n,k);
```

This thesis is intended for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    int sumA=0;
    int sumB=0;
    for(int i=0;i<input_i;i++){
     sumB=sumB+B[i][k];
    }
    for(int j=0;j<input_j;j++){
     sumA=sumA+A[j][k];
    }
    return sumA+sumB;
  }
  public void fillNi(){ //fill array by input_Ni of TFrame
    int index=0;
    TFrame tFrame=(TFrame)vectorTFrame.elementAt(index);
//    int tmpN[]=new int[input_i];
    int offset=0;
    for(int i=0;i<input_i;i++){
      NiArray[i]=tFrame.getNi(i-offset);
      if((NiArray[i]==-1)&&(index<vectorTFrame.size())){
        index++; //point to next object of TFrame
        tFrame=(TFrame)vectorTFrame.elementAt(index);//load new object
        offset+=8;
        NiArray[i]=tFrame.getNi(i-offset);
      }
    }
  }
  public void fillYarray(int i,int maxCol){
    int row,col;
    int binArray[][]=new int[i][maxCol];
    int dec,tmpDec;
 //---- make binary code ----
    dec=tmpDec=0;
    for(col=0;col<maxCol;col++){
```

This content is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    tmpDec=dec;
    for(row=0;row<i;row++){
     binArray[row][col]=tmpDec%2;
     tmpDec=tmpDec/2;
    }
    dec++;
   }
   for(col=0;col<maxCol;col++){
    for(row=i-1;row>=0;row--){
      yArray[row][col]=binArray[row][col];
    }
   }
}
public int emptyRowCell(int i){
  return input_j-NiArray[i];
}
public void fillZarray(){
 int index=0;
 for(int i=0;i<input_i;i++){
   for(int j=0;j<input_j;j++){
     zArray[i][j]=0;
   }
 }
 for(int i=0;i<input_i;i++){
   for(int m=0;m<NiArray[i];m++){
     zArray[i][m]=1;
   }
 }
 System.out.println("\n-----zArray-------");
 for(int i=0;i<input_i;i++){
  for(int j=0;j<input_j;j++){
    System.out.print("["+zArray[i][j]+"]");
```

This is for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  }
  System.out.println();
 }
 testNset(); //for first ZArray
 for(int n=0;n<input_i;n++){
  for(int m=0;m<input_j;m++){
   rotate(n);
   testNset();
System.out.println("i:="+n+ "  j:="+m);
   //++++++ rotate one row
   //for(int nn=0;nn<input_i;nn++){ //row of zArray
    //for(int mm=0;mm<input_j;mm++){
     //rotate(nn);
     //testNset();
    //}
   //}
  }//outer
 }//outer
}
public String[][][] getTotalMinC(){return totalMinC;}
public int[][] getMinValueB(){return minValueB;}
public int[][] getMinValueA(){return minValueA;}
public int[][] getMinZarray(){return minZarray;}
public String[][] getYin(){return checkMinRow(minValueB);}
public String[][][] getXijk(){return checkMinCol(totalMinC);}
public void fill(int row,int col){
 //indexdisp.setMessage("\nindex: "+count+"  ["+row+"]["+col+"]");
 count++;
 if(row==0){
 }else{
  fill(row-1,col);
 }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
 if(col==0){
   rotate(row);
   testNset();
 }else{
   fill(row,col-1);
   rotate(row);
   testNset();
 }
}
public void rotate(int row){
 int tmp1=zArray[row][0],
     tmp2=0;
 //indexdisp.setMessage("\nindex: "+count);
 //count++;
 for(int j=1;j<input_j;j++){
   tmp2=zArray[row][j];
   zArray[row][j]=tmp1;
   tmp1=tmp2;
 }
 zArray[row][0]=tmp1;
}
public void rotate(int row,int pos){
 int tmp = zArray[row][pos+1];
 zArray[row][pos+1]=zArray[row][pos];
 zArray[row][pos]=tmp;
}
public long findKmap(int j){
  long result=1;
  for(int i=0;i<input_i;i++){
    result=result*(fac(j)/(fac(j-NiArray[i])*fac(NiArray[i])));
  }
  return result;
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
}
public int fac(int n){
  if(n<=1)
    return 1;
  else
    return n*fac(n-1);
}
public int power(int b,int p){
  int result=1;
  if(p==1)return b;
  else return b*power(b,p-1);
}
}
class Disp extends Frame{
  TextArea t;
  Font f;
  Disp(String s){
    super(s);
    t=new TextArea(400,500);
    f=new Font("TimesRoman",Font.BOLD,14);
    t.setFont(f);
    add(t);
    resize(500,600);
    show();
  }
  public void setMessage(String s){
//    t.setFont(f);
    t.appendText(s);
  }
  public void clearText(){
    t.setText("");
  }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
  public boolean handleEvent(Event e){
    if(e.id==Event.WINDOW_DESTROY){
      hide();
      dispose();
      return true;
    }
    return super.handleEvent(e);
  }
}
class AppendFileStream extends OutputStream {
  RandomAccessFile fd;
  public AppendFileStream(String file) throws IOException {
    fd = new RandomAccessFile(file,"rw");
    fd.seek(fd.length());
    }
  public void close() throws IOException {
    fd.close();
    }
  public void write(byte[] b) throws IOException {
    fd.write(b);
    }
  public void write(byte[] b,int off,int len) throws IOException {
    fd.write(b,off,len);
    }
  public void write(int b) throws IOException {
    fd.write(b);
    }
  }
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

AUTHOR BIOGRAPHY

Author	Miss Prattana Maneechay
Date of Birth	July 21, 1976
Bachelor Degree	B.Sc.(Applied Statistics)
Institution	Faculty of Science and Technology, Rajabhat Institute Suan Dusit, Bangkok, Thailand
Year of Graduation	1996

King Mongkut's Institute of Technology Ladkrabang

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.