SOFTWARE REQUIREMENTS FOR A FACILITIES DESIGN SOFTWARE AND EVALUATION OF THE FACTORY PROGRAMS SUITE

A Thesis Presented to

The Faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

By
Shivaram Kota
March 2000
THIS THESIS ENTITLED

“SOFTWARE REQUIREMENTS FOR A FACILITIES DESIGN SOFTWARE AND EVALUATION OF THE FACTORY PROGRAMS SUITE”

By: Shivaram Kota

has been approved

for the Department of Industrial and Manufacturing Systems Engineering and

the Russ College of Engineering and Technology

Charles M. Parks, Ph.D., P.E.
Professor and Chairman of Industrial and Manufacturing Systems Engineering

Warren K. Wray, Dean
Fritz J. and Dolores H. Russ
College of Engineering and Technology
Acknowledgements

I would like to thank my advisor, Dr. Charles M. Parks, and my other committee members, Dr. David Koonce and Dr. Bhavin Mehta, for their help and support on my thesis. Also, I would like to thank Dr. Thomas Lacksonen for his continuous help and encouragement. Finally, I wish to thank my family for their continuing support.

Shivaram Kota
# TABLE OF CONTENTS

LIST OF TABLES ................................................. vii
LIST OF FIGURES .................................................. viii

CHAPTER 1

PROBLEM STATEMENT ........................................... 1
1.1 Literature Review ........................................... 2
1.2 Suggested Approach ......................................... 3

CHAPTER 2

METHODOLOGY OF FACILITIES DESIGN .............. 4
2.1 Location Analysis ........................................... 4
2.2 General Overall Layout ................................... 5
2.3 Detail Layout .................................................. 5
2.4 Installation ..................................................... 6

CHAPTER 3

DETERMINING THE PROXIMITY RELATIONSHIPS ... 13

CHAPTER 4

SOFTWARE REQUIREMENTS ................................. 18
4.1 Input Requirements .......................................... 18
4.2 Algorithmic Requirements ................................ 21
4.3 Library Requirements ....................................... 25
4.4 Output Requirements ........................................ 25
4.5 Decision support and Simulation System .......... 25

CHAPTER 5

THE CIMTECH SOFTWARE SUITE .......................... 29
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Determining Space Requirements</td>
<td>8</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Space Requirements Converting</td>
<td>9</td>
</tr>
<tr>
<td>Table 2.4</td>
<td>Ratio Trend Projection</td>
<td>12</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Sample report by products</td>
<td>37</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Sample material handling report</td>
<td>38</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>Flow Cost Results</td>
<td>47</td>
</tr>
<tr>
<td>Table 7.2</td>
<td>Comparison of Execution Times</td>
<td>48</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.3</td>
<td>Estimating space requirements by the use of pre-determined space standards</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Material flow intensity</td>
<td>14</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Relationship scores</td>
<td>16</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Adjacency graph</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Aisle space estimation</td>
<td>19</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Symbols developed by American Standards Association</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Symbols developed by American Standards Association</td>
<td>27</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Path description</td>
<td>34</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>Grouping problem</td>
<td>40</td>
</tr>
<tr>
<td>Figure 6.2</td>
<td>The priority algorithm</td>
<td>42</td>
</tr>
</tbody>
</table>
A well developed facilities planning software can assist industrial engineers in a number of ways. The importance of efficient factory layouts has long been recognized by the practicing engineer. A good factory layout can offer many benefits, such as reduced work-in-process inventory, reduced waiting time, higher quality, etc. There are a number of facilities design software packages which help in designing a facilities layout. Development of a set of requirements for the facilities layout software coupled with a description of the importance of each of these requirements will aid in evaluating the software and will help in analyzing, evaluating and improving the software. Currently there is no detailed requirements set developed which can be used to evaluate a facility layout software. A set of such requirements will be developed in this thesis. This set will then be used to evaluate the facilities planning software suite (FACTORYCAD, FACTORYFLOW, FACTORYPLAN/OPT) developed by Cimtechnologies, Inc.
1.1 LITERATURE REVIEW

"Systematic Layout Planning" by Richard Muther [1] is probably the most widely read book by facilities designers. Muther attempted to provide procedures to solve layout problems systematically. Many of Muther's procedures have been incorporated into computer programs to aid the facility designers in the plant layout processes. Quartermann Lee [5] has proposed a few software functions for a facilities engineering software, but his set is very limited, and most of the functions are management oriented (like selection of sites, strategic planning, etc.) but are not inclined towards the engineering aspects. As part of an NSF project, Irani and Agarwal [4] have evaluated the capabilities of FACTORYOPT, a facilities engineering software, but the evaluation was not based on a set of predefined requirements for a good facilities software. There are various software packages which help the user in developing a layout, and Imam [12] briefly discusses the functionality of each of the software packages.

The FACTORY PROGRAMS suite developed by Cimtechnologies Inc is the most comprehensive facility layout and design software currently available and it provides features which help in designing and simulating a layout. This software suite is also capable of providing numerical results to judge the efficiency of a given layout. The approach taken by the Cimtechnologies group towards facility layout and design can be obtained from the Cimtechnologies manuals [9, 10 and 11]. A priority algorithm was developed by the Department of Industrial and Manufacturing Systems Engineering at Ohio University. Preliminary results indicate that if the priority algorithm is
implemented, it could turn out to be more efficient and versatile when compared to the existing flow algorithm used by FACTORY FLOW software.

1.2 Suggested Approach

The tasks for this research are as follows:

1. Review various phases of facilities planning.

2. Develop a set of requirements that are vital for facilities planning based on the above review.

3. Analyze the Cimtech software suite based on the requirements set developed.

4. Improve the practical applicability and the scope of the Cimtech software suite by implementing and testing a priority algorithm designed to solve a specific shortcoming of the software.
CHAPTER 2

METHODOLOGY OF FACILITIES DESIGN

The overall process of facilities design can be divided into four phases, as shown below.

**Phase 1**
Location analysis: determine the location of the area to be laid.

**Phase 2**
General Overall Layout: establish the general arrangement of the area to be laid out.

**Phase 3**
Detailed layout plans: locate each specific piece of machinery and equipment.

**Phase 4**
Installation: plan the installation, seek the approval of the plan, make the necessary physical moves.

2.1 Location Analysis

This is a determination of what is actually required to correct problems or meet new challenges. Location analysis may involve a new site, but more often it involves redesigning the present location, rearranging several locations within the present plant, or arranging a newly acquired site or some other available space. Sometimes the layout planner may not be involved during these early decisions particularly when a new site is being considered. If a new site is being considered, then a short list of various possible sites should be developed. Various aspects like those described below should be considered before making a decision regarding the site.
- Availability of local trucking services.
- Specifics regarding short and long term capacity availability and reliability for water, sewer, gas, electricity and the area’s power outage history.
- Total taxing differences among alternative sites.
- The quality of life differentials, real estate price, etc.

As we can see from the above description, most of the decisions involved in this phase involve various qualitative factors and other categorical variables. The people involved in doing this selection have to think as far into the future as possible. Software has limited use in such decisions.

2.2 General Overall Layout

This phase is also referred to as the block layout phase. This phase determines the basic flow patterns and major individual area allocations. It provides the general size and configuration of each major area and proximity relationships, affinities, and major material flows between these areas. Main aisles and aisle patterns should be included in this phase.

2.3 Detail Layout

Detail layout planning follows the same general pattern of procedures as block layout planning. The level of detail and information, particularly on physical plant and equipment specifications, is at a fairly high level. The mechanical process or logic used in detail layout planning within specific departments or work cells is very similar to the process used to develop block layouts. Instead of the relationships being developed
between blocked out areas, they are now developed between individual operations and/or machines. The detail layout of a production area should show each machine, each operator, materials set down area, materials pickup area, auxiliary equipment, benches, aisles, etc. Library parts preprogrammed in the CAD system for individual pieces of equipment become very useful at this point.

2.4 Installation

In this phase, installation instructions and time line schedules are developed, and the various approvals, appropriations, and permits needed to perform the actual installation are secured. Typically, a major facet of this phase is scheduling and implementing the rearrangement or move with the least amount of disruption to ongoing production operations. The key factors according to [3] in implementation planning are project management and good communication.

As we can see from the above description, the first and the last phases are highly qualitative in nature and management oriented with virtually no engineering tasks (decision support system is sufficient in these two stages).

The basic requisites needed to develop a facilities design software to design a facilities layout are discussed in the rest of the chapter.

A software system to assist the facilities designer is helpful in the second and third stages. The stages in the block layout phase can be split into the following phases:

1. Space calculations
2. Proximity relationships and affinities.
Space calculations: There are five different ways of determining the space requirements. These are:

a) Calculations
b) Converting
c) Space standards
d) Roughed-out layout
e) Ratio trend and projection

Calculations: This method of determining space requirements involves breaking down each area or activity into the subareas and individual space elements that make up its total space. Then the amount of area for each space element is determined and then multiplied by the number of elements required to do the job, as shown in Table 2.1

Converting: This is also called the factoring method (Table 2.2) which establishes what space is being occupied at the present time and converts it to what will be required for the proposed layout. This conversion is usually a matter of logic, best estimate, or educated guess.

Space Standards: Pre-established space standards (Figure 2.3) are used in this method to determine requirements for projects. Using this method should be case dependent. Various other factors, such as the working conditions, have to be considered or anticipated while determining the space needs. Once a company develops its own standards and conditions, they could be used repetitively.
### Table 2.1 Determining Space Requirements

From: Muther [1]

<table>
<thead>
<tr>
<th>Activity — Area of Dept.</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
<td>Sq. Ft</td>
</tr>
<tr>
<td><strong>Purchased Parts Storage</strong></td>
<td>30</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>75</td>
<td>+20</td>
<td>120</td>
<td>120</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Machine Shop</strong></td>
<td>400</td>
<td>1000</td>
<td>-15</td>
<td>850</td>
<td>850</td>
<td>+10</td>
<td>1100</td>
<td>1050</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Brazing</strong></td>
<td>60</td>
<td>120</td>
<td>+50</td>
<td>180</td>
<td>150</td>
<td>+75</td>
<td>210</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Machining</strong></td>
<td>50</td>
<td>100</td>
<td>+20</td>
<td>120</td>
<td>110</td>
<td>+50</td>
<td>150</td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Plating</strong></td>
<td>40</td>
<td>150</td>
<td>+10</td>
<td>165</td>
<td>165</td>
<td>+30</td>
<td>195</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Welding</strong></td>
<td>190</td>
<td>400</td>
<td>-5</td>
<td>380</td>
<td>380</td>
<td>+10</td>
<td>440</td>
<td>440</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sand Blast</strong></td>
<td>45</td>
<td>90</td>
<td>+5</td>
<td>95</td>
<td>90</td>
<td>+10</td>
<td>100</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Seal Making</strong></td>
<td>190</td>
<td>400</td>
<td>+20</td>
<td>480</td>
<td>440</td>
<td>+30</td>
<td>520</td>
<td>500</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Leak Test</strong></td>
<td>20</td>
<td>40</td>
<td>+25</td>
<td>50</td>
<td>50</td>
<td>+30</td>
<td>50</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vacuum Exaustion</strong></td>
<td>25</td>
<td>100</td>
<td>+10</td>
<td>110</td>
<td>100</td>
<td>+50</td>
<td>150</td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Gas Preparation</strong></td>
<td>165</td>
<td>370</td>
<td>+20</td>
<td>445</td>
<td>400</td>
<td>+25</td>
<td>460</td>
<td>450</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Receiving</strong></td>
<td>150</td>
<td>150</td>
<td>+10</td>
<td>165</td>
<td>165</td>
<td>+20</td>
<td>180</td>
<td>180</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Shipping</strong></td>
<td>200</td>
<td>200</td>
<td>+10</td>
<td>220</td>
<td>200</td>
<td>+20</td>
<td>240</td>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Office and Frest Rooms</strong></td>
<td>300</td>
<td>280</td>
<td>0</td>
<td>280</td>
<td>260</td>
<td>+30</td>
<td>365</td>
<td>300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TOTALS** | 1880 | +1670 | 3550 | — | 3705 | 3500 | — | 4350 | 4200 |
### Table 2.2 Space Requirements converting

**From: Muther [1]**

<table>
<thead>
<tr>
<th>Machine or Equipment Identification Number</th>
<th>Identification Data (Name and/or Description)</th>
<th>Space</th>
<th>Physical Features Required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left - Right inches</td>
<td>Front - Back inches</td>
<td>Height</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>Area Total Brought Forward</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bake Area</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-1 Sink</td>
<td>24 25 - 4 6 4</td>
<td>14 16 - 14 16</td>
<td>14 16</td>
</tr>
<tr>
<td>1-2 Food Prep. Table</td>
<td>19 20 11 - 24 11 1</td>
<td>18 20 11 - 14 11 1</td>
<td>14 11</td>
</tr>
<tr>
<td>1-3 Steamer (Step)</td>
<td>10 10 10 - 4 4 1</td>
<td>8 8 - 8 8 1</td>
<td>8 8 1</td>
</tr>
<tr>
<td>1-4 Proof Box</td>
<td>40 40 10 - 16 10 1</td>
<td>7 7 10 - 7 10 1</td>
<td>7 10 1</td>
</tr>
<tr>
<td>1-5 Bag Mixers</td>
<td>10 40 - 10 40</td>
<td>10 40 - 10 40</td>
<td>10 40</td>
</tr>
<tr>
<td>1-6 Attack Mixer, Hobart A202</td>
<td>16 18 - 21 5 1 4 1</td>
<td>11 11 11 4 1</td>
<td>11 11</td>
</tr>
<tr>
<td><strong>Cook Area</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-7 Range</td>
<td>21 6 4 7 12 1</td>
<td>7 12 1</td>
<td>7 12 1</td>
</tr>
<tr>
<td>1-8 Bottle</td>
<td>22 14 14 14</td>
<td>7 10 1</td>
<td>7 10 1</td>
</tr>
<tr>
<td>1-9 Portable Shef, Model #800</td>
<td>26 26 26 26 - 16 1</td>
<td>10 10 10 10 1</td>
<td>10 10</td>
</tr>
</tbody>
</table>

* Required space for main or delivery aisles and service areas not included.

**Reference Notes:**

1. Door Swing is 21 1/2 inches
2. Door Swing is 18 inches
3. Door Swing is 18 inches
4. Door Swing is 18 inches
5. Door Swing is 18 inches
6. Total Area Pa'd. **2,430**

**Machinery & Equipment Area & Features Sheet**

- **Company/Plant**: North
- **Bldg/Dept/Area**: Cafeteria-Kitchen
- **Project**: 2.35.3
- **By**: O.C. Gomer
- **With**: D.F. Adams
- **Date**: 11-1-3
- **Sheet**: 2 of 2
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Min. Area Required (square feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR-1-B</td>
<td>B &amp; D 10&quot; Bench</td>
<td>4</td>
</tr>
<tr>
<td>GR-1-B</td>
<td>B &amp; D 6&quot; Pedestal</td>
<td>4</td>
</tr>
<tr>
<td>GR-1-B</td>
<td>Black &amp; Decker H-4700</td>
<td>7</td>
</tr>
<tr>
<td>GR-1-B</td>
<td>Blount 1½&quot; Heavy Duty</td>
<td>18</td>
</tr>
<tr>
<td>GR-1-C</td>
<td>Cinn. Pedestal</td>
<td>7</td>
</tr>
<tr>
<td>GR-1-S</td>
<td>Sterling 14&quot; Tool</td>
<td>18</td>
</tr>
<tr>
<td>GR-1-S</td>
<td>Standard</td>
<td>15</td>
</tr>
<tr>
<td>GR-1-V</td>
<td>Van Norman 39 Spcl.Rad.</td>
<td>45</td>
</tr>
<tr>
<td>GR-1-W</td>
<td>Wickes 2-Spdl. Tool</td>
<td>12</td>
</tr>
<tr>
<td>GR-2-B</td>
<td>Badger 2:0</td>
<td>64</td>
</tr>
<tr>
<td>GR-2-H</td>
<td>Hanchett 121</td>
<td>100</td>
</tr>
<tr>
<td>GR-3-G</td>
<td>Gardner 125</td>
<td>120</td>
</tr>
<tr>
<td>GR-3-H</td>
<td>Haas 230</td>
<td></td>
</tr>
</tbody>
</table>

Fig 2.3 Estimating space requirements by use of pre-determined space standards
From: Muther [1]
Roughed-Out-layout: When a scale plan of the area and templates or models of the equipment involved are already on hand and if certain activities are critical or represent a very high investment then this type of approach is recommended. This method of space determination is expected to be used for critical areas of high investment, relatively fixed equipment, or multiple workstations that should line up.

Ratio trend projection: This method is considered to be the least accurate of all the five methods of calculating space requirements. The ratio trend projection method is limited to general space requirements. The ratio trend and projection method establishes a ratio of square feet to some other factor, square feet per person employed. First each of such ratios in various periods of time in the past is established and then the likely ratio for the future is projected (Refer to Table 2.4).

To accomplish the above tasks, a good CAD system supported by a DSS would suffice.

Phase 2: Proximity relationships and affinities

In this phase, types (material or nonmaterial) of flows between various departments are identified. This task is generally done based on experience and the basic objective of building the layout. After establishing the flows, the proximity relationships between various departments are determined.
<table>
<thead>
<tr>
<th></th>
<th>4 Yrs. Ago</th>
<th>2 Yrs. Ago</th>
<th>This Year</th>
<th>Two Years From Now</th>
<th>Five Years From Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$ Net Sales</td>
<td>385M</td>
<td>855M</td>
<td>1,300M</td>
<td>1,800M</td>
</tr>
<tr>
<td>2</td>
<td>No. of pieces (units) produced</td>
<td>805</td>
<td>1,720</td>
<td>2,660</td>
<td>3,800</td>
</tr>
<tr>
<td>3</td>
<td>No. of Shop Employees</td>
<td>15</td>
<td>35</td>
<td>51</td>
<td>(1 ÷ 7) 67</td>
</tr>
<tr>
<td>4</td>
<td>No. of Office Employees</td>
<td>8</td>
<td>11</td>
<td>16</td>
<td>(1 ÷ 8) 20</td>
</tr>
<tr>
<td>5</td>
<td>Square feet in Service Shop</td>
<td>5,250</td>
<td>11,000</td>
<td>17,000</td>
<td>(1 ÷ 9) 22,500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(3 x 11) 23,400</td>
<td>(3 x 11) 32,600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(2 x 13) 24,700</td>
<td>(2 x 13) 35,800</td>
</tr>
<tr>
<td>6</td>
<td>Square feet in Office</td>
<td>750</td>
<td>1,600</td>
<td>2,100</td>
<td>(1 ÷ 10) 2.840</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(4 x 12) 2.700</td>
<td>(4 x 12) 3,200</td>
</tr>
<tr>
<td>7</td>
<td>$ Net Sales per Employee (Shop)</td>
<td>$25,600</td>
<td>$24,400</td>
<td>$25,500</td>
<td>27,000</td>
</tr>
<tr>
<td>8</td>
<td>$ Net Sales per Employee (Office)</td>
<td>48,200</td>
<td>77,600</td>
<td>81,300</td>
<td>90,000</td>
</tr>
<tr>
<td>9</td>
<td>$ Net Sales per sq. ft. (Shop)</td>
<td>73</td>
<td>78</td>
<td>77</td>
<td>80</td>
</tr>
<tr>
<td>10</td>
<td>$ Net Sales per sq. ft. (Office)</td>
<td>514</td>
<td>535</td>
<td>620</td>
<td>635</td>
</tr>
<tr>
<td>11</td>
<td>Sq. ft. Shop per Shop Employee</td>
<td>350</td>
<td>315</td>
<td>334</td>
<td>350</td>
</tr>
<tr>
<td>12</td>
<td>Sq. ft. Office per Office Employee</td>
<td>94</td>
<td>145</td>
<td>131</td>
<td>135</td>
</tr>
<tr>
<td>13</td>
<td>Sq. ft. Shop per piece (unit) produced</td>
<td>6.5</td>
<td>6.4</td>
<td>6.4</td>
<td>6.5</td>
</tr>
</tbody>
</table>

A. Taken from company records or estimates of officials.
B. Ratios derived from figures in "A".
C. Forecast sales and production.
D. Ratios projected—based on trends and anticipated products and methods.
E. Manpower calculations based on "C" forecasts and "D" ratios.
F. Square foot calculations based on "C" and "E" figures, and "D" ratios.
CHAPTER 3

DETERMINING THE PROXIMITY RELATIONSHIPS

One of the most important factors to be considered while designing a facilities layout is to determine the placement of the departments based on the interaction between them.

Relationships among departments can be determined in two ways: a) Quantitatively or b) Qualitatively. Quantitative analysis is based on the material flow analysis between the departments. Material flow could mean anything like the actual movement of raw material, finished products, material handling equipment, etc. An example is shown in Figure 3.1. The intensity of material flow between departments can be classified into:

- **A** = Abnormally high intensity of flow

  These departments may need to be close to each other since the interface between these departments is very high. E.g: the relationship between the incoming inspection department and the receiving docs may need an A if there is a high level of interaction. Also, the departments might have to be close for security reasons.

- **E** = Especially high intensity of flow

  This grade is assigned when two departments can exist without being adjacent to each other, but many benefits will accrue if they are placed together. E.g: putting the materials receiving area close to the materials storage area.

- **I** = Important intensity of flow

  There would not be any drastic effect if these functions are not adjacent, but the operation would be a little smoother if these functions are placed together; e.g: placing the clerical
Fig 3.1 Material flow intensity
From: Phillips [3]
staff away from the managers cabin would not be as detrimental to the plant operations as separating crucial production functions.

\[
O = \text{Ordinary intensity of flow}
\]

Separating two functions will not prove as costly as in first three cases mentioned above.

\[
U = \text{Unimportant moves of negligible intensity}
\]

\[
X = \text{Should never be together.}
\]

E.g: Presence of explosive materials adjacent to a human work place.

Qualitative analysis of the closeness ratings between departments is based on the non-flow factors between the departments. Typical non-flow relationships that could be considered while designing a layout are:

- Maintenance
- Tool storage
- Lunchrooms
- Restrooms
- Quality assurance
- General offices
- Locker rooms

These relationships are ranked again according to the A, E, I, O, U, X principle.

Scores are then established for each of the alphabetical ratings and an aggregated score is derived from these two scores (Figure 3.2). Based on this aggregated score, adjacencies (Figure 3.3) are determined.
Using the preceding layout engineering aspects, a set of software requirements were developed. These requirements will be discussed in the ensuing chapter.

<table>
<thead>
<tr>
<th>Activity pairs</th>
<th>Flow intensity</th>
<th>Conventional scores</th>
<th>Weight factors</th>
<th>Nonflow ratings</th>
<th>Nonflow scores</th>
<th>Combined scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-5</td>
<td>E</td>
<td>3</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>3-11</td>
<td>O</td>
<td>1</td>
<td>2</td>
<td>E</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5-7</td>
<td>U</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7-9</td>
<td>A</td>
<td>4</td>
<td>8</td>
<td>I</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2-4</td>
<td>A</td>
<td>4</td>
<td>8</td>
<td>E</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>3-4</td>
<td>E</td>
<td>3</td>
<td>6</td>
<td>O</td>
<td>-</td>
<td>7</td>
</tr>
<tr>
<td>4-9</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>2-3</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td>A</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 3.2 Relationship Scores
From Phillips [3]
Figure 3.3 Adjacency Graph
From: Phillips [3]
CHAPTER 4

SOFTWARE REQUIREMENTS

Designing a facilities layout requires a wide range of considerations, and therefore a corresponding set of requirements in a facilities design software can help the designer produce better layouts. A set of such requirements is presented below. The software requirements are classified into the following groups:

1. Input requirements
2. Algorithmic requirements
3. Library requirements
4. Output requirements
5. Simulation system

4.1 Input Requirements:

1. A CAD tool suitable to draw a facilities layout.

   A good CAD tool must be able to provide a library of standard mechanical, electrical and other industrial equipment such as cranes, conveyors racks etc. The factory CAD module of the Cimtech software suite is an excellent drafting tool.

2. Special feature to accommodate aisles in the layout

   According to [3], depending on the overall size of the facility, it is common for the main isles to occupy up to 10 - 18% of the total under roof floor space. Figure 4.1 illustrates this aspect. This percentage is exclusive of the isles inside the departments
or cells. So if the aisle space is not considered, then the planner could underestimate the space requirement for the layout.

Aisle space: 1.4%
Gross space = 73,984 ft²
Net space = 65,536 ft²

Fig 4.1 Aisle space estimation
From: Phillips [3]
3. Allowance for scrap factor in flow analysis.

Consideration of scrap factor during production analysis is useful to generate meaningful results.

4. Ability to draw areas of any shape

Since the presence of non-rectangular departments is evident in the industry, the facilities design software should have the provision for the designer to be able to draw areas of any shape.

5. Ability to accept factory operating characteristics like part flows, product, part, machine and material handling equipment information.

6. Provision to consider equipment utilization during space planning.

According to[3], it is very important to use the correct utilization factors in space planning. Errors in utilization would lead to incorrect calculation of space requirements and will form the root of problems for a long time. For example, if 10 machines are needed to manufacture 100 parts an hour and if each machine occupies 10 sq.m of area then the space needed would be 100 sq.m. This is correct if 100% utilization of the machine is assumed. But 100% utilization is hypothetical. So if the utilization is around 80% then 12.5 machines are needed for the same output and the corresponding space requirement increases to 125 sq.m.

It is also important to consider machine breakdowns and the repair times of each of the machines to increase the accuracy of the results produced.
4.2 Algorithmic Requirements

1. Determine the closeness ratings among various interacting departments.

One of the most important factors to be considered while designing a facilities layout is to determine the placement of the departments based on the interaction between them. Relationships between departments can be determined in two ways -- quantitatively or qualitatively. Quantitative analysis is based on the material flow analysis between the departments. Material flow could mean anything such as the actual movement of raw material, finished products, material handling equipment, etc.

The intensity of material flow between can be classified into:

- **A** = Abnormally high intensity of flow
- **E** = Especially high intensity of flow
- **I** = Important intensity of flow
- **O** = Ordinary intensity of flow
- **U** = Unimportant moves of negligible intensity

Qualitative analysis of the closeness ratings between departments is based on the non-flow factors between the departments. Typical non-flow relationships that could be considered while designing a layout are:

- Maintenance
- Tool storage
- Lunchrooms
- Restrooms
- Quality assurance
- General offices
- Locker rooms

These relationships are ranked again according to the A, E, I, O, U principle.

Scores are then established for each of the alphabetical ratings and an aggregated score is derived from these two scores.

2. Provision of a good process planning software

A process planning tool would be very helpful in knowing the sequence of operations to be performed on a product based on the features it has. This could greatly help in determining the affinities between departments based on the sequence of flow of the product. This would also help in picking the proper machining tool and material handling equipment.

3. Design of cellular manufacturing systems.

Group technology refers to the grouping of parts into families and then making design decisions based on family characteristics. Groupings are typically based on part shapes, part sizes, material types, and process requirements. The use of group technology can have positive internal impacts in several areas. A few of these impacts are listed below:

a) Frees up engineering resources to do other things.

b) Lowers overall inventories and reduces storage space needs.

c) Increases equipment utilization.
d) Simplifies materials purchasing, control, and scheduling procedures

e) Could help to simplify and streamline shop floor layouts.

Algorithms like single-pass heuristic [6] and similarity coefficients [6] could be embedded in the software to determine the grouping. Single-pass heuristic is designed for the case where it is desired to totally eliminate all inter group moves. In the similarity coefficients method, emphasis is on locating machines in the same group. So the choice of algorithm should be at the user's discretion.

4. Material flow and material handling analysis

   A well designed material handling system would be very beneficial.

   a) Generates savings in ongoing materials handling transport costs.

   b) Can help lower inventory and production control costs.

   c) Can increase production capacity

   d) Can provide better space utilization

   e) Can reduce parts damage and waste

   f) Can help to implement shorter production time cycle

   g) Can provide a high degree of safety

5. Provision to automatically select the material handling equipment based on cost and load factors.

   The choice of material handling equipment depends on unit loads, travel distances, travel paths, handling costs, etc. A good estimation of the cost of travel between departments is very important for an efficient layout.

Presence of multi floor layouts is realistic in the industry. So a good facilities layout design software should have the ability to design and perform adjacency on multifloor layouts.

7. Provision for design of layouts containing departments of unequal areas

The existence of departments with unequal areas is very common in the industry. So a factory plan should allow the user to input departments of equal and unequal areas use appropriate algorithms to generate a near optimal adjacency solution.

8. Application of appropriate algorithms while performing adjacency analysis of departments in a layout.

9. Ability to use (for various calculations) the actual path of material flow.

10. Ability to perform congestion analysis in aisles.

It would be very helpful if congestion analysis could be performed by the facilities design software. It could help determine the busiest routes (in terms of the movement of material handling equipment or product flows), and in this process alternate routes could be identified to facilitate efficient operation of the plant.

11. Ability to perform flow analysis in multifloor layouts.

Since it is realistic to have multifloor layouts, it is essential for a facilities design software to be able to analyze material flow in such cases. Layout algorithm developed by Bozer and Meller[8] could be used to analyze multifloor layouts.
4.3 **Library Requirements**

1. Provision of a library of symbols for industrial equipment (conveyors, cranes, furniture, etc.) which could be used while drawing the layout (Figure 4.2).

2. Provision for those standard symbols for plumbing, piping, valves and electrical equipment developed by American Standards Association (Figure 4.3).

4.4 **Output Requirements**

1. Ability to generate comparative scores on alternative layouts.

   The facilities design software should have the provision to generate and edit an adjacency graph based on adjacency specifications. The design software should also be able to generate a score on relative efficiencies of various alternatives of a given layout.

2. Ability to generate reports on material handling equipment, cost of manufacturing a product etc.

4.5 **Decision Support and Simulation System**

Provision of a decision support system: A decision support system is useful in analyzing the input data needed to generate a good plant layout. Some of the most common data needed to generate a good plant layout are listed below:

   a) Configuration of the product being manufactured.

   b) Quantity of each product being manufactured.

   c) Operations process charts showing the equipment cycle times and routings.

   d) Configuration of the product being manufactured.
**Figure 4.2 Symbols developed by American Standards Association**

From: Muther [1]
Although there is no universal set of standard installation symbols used by layout planners, the symbols on the two pages of this appendix are probably the most used and accepted. Source: *Time-Saver Standards, An Architectural Record Book*, F. W. Dodge Corporation, New York, N. Y. Each company should agree upon and formalize in writing its own code of symbols based on its present practice; these standards, what is available in pre-printed tapes or sheet form, and the like.

---

### ELECTRICAL SYMBOLS FOR ARCHITECTURAL PLANS

*Prepared by American Standards Association and published by Industry Committee on Interior Wiring Design*

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
<th>Description</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GENERAL OUTLETS</strong></td>
<td></td>
<td><strong>SWITCHES</strong></td>
<td></td>
</tr>
<tr>
<td>Outlet</td>
<td></td>
<td>Single Pole Switch</td>
<td>$</td>
</tr>
<tr>
<td>Capped Outlet</td>
<td></td>
<td>Double Pole Switch</td>
<td>$s</td>
</tr>
<tr>
<td>Deep Cord</td>
<td></td>
<td>Three Way Switch</td>
<td>$s_3</td>
</tr>
<tr>
<td>Electrical Outlet</td>
<td></td>
<td>Four Way Switch</td>
<td>$s_4</td>
</tr>
<tr>
<td>Fan Outlet</td>
<td></td>
<td>Automatic Door Switch</td>
<td>$s_0</td>
</tr>
<tr>
<td>Junction Box</td>
<td></td>
<td>Electrolier Switch</td>
<td>$s_e</td>
</tr>
<tr>
<td>Lamp Holder</td>
<td></td>
<td>Key Operated Switch</td>
<td>$S_k</td>
</tr>
<tr>
<td>Lamp Holder with Pull Switch</td>
<td></td>
<td>Switch and Pilot Lamp</td>
<td>$S_p</td>
</tr>
<tr>
<td>Lampholder</td>
<td></td>
<td>Circuit Breaker</td>
<td>$S_cb</td>
</tr>
<tr>
<td>Lamp Holder with Pull Switch</td>
<td></td>
<td>Weatherproof Circuit Breaker</td>
<td>$S_{bc}</td>
</tr>
<tr>
<td>Pull Switch</td>
<td></td>
<td>Momentary Contact Switch</td>
<td>$S_{mc}</td>
</tr>
<tr>
<td>Outlet for Vapor Discharge Lamp</td>
<td></td>
<td>Remote Control Switch</td>
<td>$S_{rc}</td>
</tr>
<tr>
<td>Exit Light Outlet</td>
<td></td>
<td>Weatherproof Switch</td>
<td>$S_{wp}</td>
</tr>
</tbody>
</table>

**SPECIAL OUTLETS**

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit Light Outlet</td>
<td></td>
</tr>
<tr>
<td>Clock Outlet (lighting only)</td>
<td></td>
</tr>
</tbody>
</table>

**CONVENIENCE OUTLETS**

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplex Convenience Outlet</td>
<td></td>
</tr>
<tr>
<td>Convenience Outlet other than Duplex</td>
<td></td>
</tr>
<tr>
<td>1 or Single, 3 or Triplex, etc.</td>
<td></td>
</tr>
<tr>
<td>Weatherproof Conv. Outlet</td>
<td></td>
</tr>
<tr>
<td>Range Outlets</td>
<td></td>
</tr>
<tr>
<td>Switch and Convenience Outlet</td>
<td></td>
</tr>
<tr>
<td>Radio and Convenience Outlet</td>
<td></td>
</tr>
<tr>
<td>Special Purpose Outlet (incl. in 27)</td>
<td></td>
</tr>
<tr>
<td>Floor Outlet</td>
<td></td>
</tr>
</tbody>
</table>

**PANELS, CIRCUITS, MISC.**

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lighting Panel</td>
<td></td>
</tr>
<tr>
<td>Power Panel</td>
<td></td>
</tr>
<tr>
<td><em>Branch Circuit—Ceiling—Wall</em></td>
<td></td>
</tr>
<tr>
<td><em>Branch Circuit—Floor</em></td>
<td></td>
</tr>
<tr>
<td>2-outlet</td>
<td></td>
</tr>
<tr>
<td>3-outlet</td>
<td></td>
</tr>
<tr>
<td>Switch</td>
<td></td>
</tr>
<tr>
<td>Feeder</td>
<td></td>
</tr>
<tr>
<td>Feeders, use heavy lines and designate by number for quick reference</td>
<td></td>
</tr>
<tr>
<td>Special Auxiliary Outlets</td>
<td></td>
</tr>
<tr>
<td>Underfloor Duct &amp; Junction Box</td>
<td></td>
</tr>
<tr>
<td>Underfloor Duct &amp; Junction Box</td>
<td></td>
</tr>
<tr>
<td>Interconnection Box</td>
<td></td>
</tr>
<tr>
<td>Battery</td>
<td></td>
</tr>
<tr>
<td>Auxiliary System Circuits (without other designation)</td>
<td></td>
</tr>
<tr>
<td>Auxiliary System Circuits (without other designation)</td>
<td></td>
</tr>
<tr>
<td>Auxiliary System Circuits (without other designation)</td>
<td></td>
</tr>
</tbody>
</table>

---

Figure 4.3 Symbols developed by american standards association.  
From: Muther [1]
e) Supporting service requirements (Space not covered by the manufacturing activities).

f) Equipment utilization

g) Total space requirements

h) Information on order characteristics (Like order size, lotsizes etc)

i) Production equipment listing and scaled drawings of each piece.

Based on the above data, a decision support system is needed to calculate some important requirements, such as the space needs based on type of product and the quantity being produced. A decision support system is also needed for statistical tabulations and calculations, product quantity plotting/charting, finance modeling, etc, needed to make some important corporate decisions regarding the layout. The system on the whole could then be simulated to see the impact of various variables on the output.
CHAPTER 5
THE CIMTECH SOFTWARE SUITE

5.1 Description of the Cimtech Software Suite

Cimtech Software suite is basically categorized into three modules. They are as follows:

1. Factory Cad: Factory Cad is a tool used for designing and drafting industrial facilities. It is used to create new drawings and refine or modify the existing drawings.

2. Factory Flow: Factory Flow is a layout analysis tool. It integrates the facilities drawings and material flow paths with production and material handling data. Text reports on various aspects like costs of using material handling equipment and utilization can be generated for detailed inspection.

3. Factory plan/Opt: Factory Plan is a tool to design and analyze layouts based on how desirable it is that activities be close to each other as determined by an assigned proximity value, the intensity of material flow between them, or an aggregate of proximity and flow relationships.

5.2 Analysis of the Cimtech Software Suite

In this Section, the Factory programs suite of Cimtechnologies Corporation will be evaluated based on the software requirements set developed previously.

5.2.1 Input Requirements

1. A CAD tool suitable to draw a facilities layout.

Factory Cad module of the Cimtech software suite is an efficient drawing tool to draft
a facilities layout. It includes a library of standard mechanical, electrical, and pre-drawn parametric symbols. It can also automatically draw industrial equipment, such as cranes, racks and conveyors to conform to user specifications.

2. Special feature to accommodate aisles in the layout.

Factory Cad module [9] has a special feature to accommodate aisles while drafting a layout. The bay aisle generator dialog specifies the parameter for drawing an aisle border.

3. Allowance for scrap factor in flow analysis.

The user can enter the scrap rate through the parts file. If the factor in the scrap rates check box in the calculate [10] dialog is checked, FactoryFlow will increase the production volumes so that the specified production quantity is achieved.

4. Ability to draw areas of any shape

The Factory Cad module[9] of the Cimtech software suite provides the user with this ability.

5. Ability to accept factory operating characteristics like part flows, product, part, machine and material handling equipment information.

The input of such characteristics can be given through the parts file(.PRT extension), products file(.PRD extension), path file(.PTH extension) and material handling equipment file(.MHE extension). The format of these input files is shown in Appendix 2.

6. Provision to consider equipment utilization during space planning.

The user has to manually calculate the equipment utilization while calculating
space requirements. Cimtech software does not have a special input box where the user could enter the utilization factor. This would be a useful feature to add since it provides immunity to manual errors in calculating space requirements for equipment being used.

5.2.2 Algorithmic Requirements

1. Determine the closeness ratings among various interacting departments.

   The Factory Plan [11] module of the Cimtech software suite achieves this objective. Data files identifying activities and their space requirements along with relationship between activities (A, E, I, O, U, X) are created in the Factory Plan editor. Using this information Factory Plan generates an activity relationship diagram. The A relationships are joined by red lines, E by yellow, I by green and O by blue. This color convention helps the user to visually identify the relationships between the departments in the diagram generated. Factory Flow allows the user to consider qualitative and flow factors and, also, to aggregate them while generating a score for the layout plan. The user has the provision to modify the generated configuration and score the layout again and compare it with the previous best score.

2. A process planning software

   The software suite does not have the planning ability in the current version.

3. Design of cellular manufacturing systems.

   There is no provision in factory flow to design cellular manufacturing systems. There is a DEFINE GROUPS(DEFGRP) option in the Adv Op menu of [10] which allows the user to define groups of machines into a group of single workcenter.
But this grouping is not the same as group technology groups for cellular manufacturing. There is no algorithmic implementation of defining these groups as done in group technology. Inability to route products through more than four consecutive groups of workcenters is a severe drawback of the Factory Flow module. It is very common to have routings through more than four groups of workcenters in the industry. This problem will be addressed in more detail at a later stage.

4. Material flow and material handling analysis.

Factory Flow module has the provision to perform material flow and material handling analysis. Some of the outputs from the factory flow module are composite flow diagram, product quantity chart, total material handling cost, material handling utilization report. Composite flow diagram visually describes the relative total cost or intensity of moving materials between pairs of workcenters.

5. Provision to automatically select the material handling equipment based on cost and load factors.

The Factory programs suite currently does not have this important provision.


The Factory programs suite lacks this provision. Implementation of algorithm developed by [8] could help in design and analysis of multifloor layouts. According to [7], LAYOPT (Facilities design software) uses the above mentioned algorithm to solve the multiple layout problem.

7. Provision for design of layouts containing departments of unequal areas

The Factory Programs suite allows the user to design and analyze departments
with equal and unequal areas. It, however does not employ a proper algorithm i.e QAP[7] to analyze a facility in which all departments have equal areas.

8. Provision of appropriate algorithms while performing adjacency analysis of departments in a layout.

   Factory Opt module provides the user a choice to generate either a Block Layout[6] or a Graph Layout[6]. It uses the two and three exchange mechanisms developed by Tompkins and White[2] to generate a Layout in both the cases. According to [2], in the process of implementing these exchange methods, the original department shapes had to be altered significantly in order to satisfy the requirements of the adjacency graph. It is left to the planner to judge the impact of this change in shape on the activities of the facility layout. According to Tompkins and White [2], one may not have as much latitude in making such alterations since department shapes are generally derived from the geometry of the individual machines within the department and the internal layout configuration. One of the restrictions of the Factory Opt module is that it does not have the provision to implement the Quadratic Assignment Problem[7] when all the departments in the layout have equal areas.

9. Ability to use (for various calculations) the actual path of material handling equipment during material movement.

   Factory flow module takes this aspect into consideration. It provides the user a choice of path types namely:

   a) Actual : The actual path taken by the material handling equipment.
b) Rectilinear: Sum of vertical and horizontal distances between two points.

c) Euclidian: Straight line path between two points.

Figure 5.1 illustrates the meanings of the above path types.

10. Ability to perform congestion analysis in aisles.

Factory Flow has the provision to perform aisle congestion analysis. The Congest calc [10] command calculates the congestion along an aisle segment and draws colored lines along those segments indicating the level of congestion. The order of intensity represented by the colored lines. Red is used for most intense route,

Yellow (next level), green (intensity lesser than yellow colored lines) and blue (least intense). The level of intensity could be displayed according to material flow intensity or material handling cost. Another option to perform congestion analysis based on the traffic of material handling equipment in aisles would be useful in properly timing the movement of material handling equipment on the shop floor and thus the efficiency.
5.2.3 Library requirements:

1. Provision of a library of symbols for industrial equipment (conveyors, cranes, furniture etc) which could be used while drawing the layout.

   Factory Cad module of the Cimtech software suite has an extensive library of such symbols.

2. Provision for those standard symbols for plumbing, piping, valves and electrical equipment developed by American Standards Association.

   Factory Cad module of the Cimtech software suite has an extensive library of such symbols.

5.2.4 Output Requirements

1. Ability to generate comparative scores on alternative layouts.

   Factory Opt module generates comparative scores on alternative layouts. 
   
   Score graph option in the optimize menu of the Factory Opt module scores a layout and generates a dialog box displaying the current score, previous highest score and percentage improvement.

2. Ability to generate reports on material handling equipment, cost of manufacturing a product etc.

   FactoryFlow generates extensive reports on material handling utilization. The report includes information, such as distance between various workcenters and the expenses involved in using the material handling equipment between the two workcenters. Reports can be created in a number of formats. Two such sample reports generated by
the Cimtech software suite are shown in Tables 5.2 and 5.3. The format desired by the user can be specified at design time.

**5.3 Decision Support and Simulation System**

Factory Flow comes with very useful analytical capabilities. Data input required by Factory Flow includes production volume, part routings, and material handling equipment data. It generates visual information and reports in text format regarding critical paths, potential flow bottlenecks, workcenter utilization, etc. Factory Flow also has the provision to generate customized reports as required by the user. Factory Flow also allows the user to make changes to the layout, routings, production volumes and other input data and examine their impact on material handling cost. Typical outputs from the Factory Flow software are total material handling cost, material handling report, material handling utilization report, workcenter utilization report product quantity chart, distance intensity chart, composite flow diagram and product flow diagram. With all these features in Factory flow, it can be concluded that Factory Flow provides a good decision support system.

The next chapter will discuss the grouping problem mentioned in requirement 3 of the algorithmic requirements section in this chapter. Though not a part of problem statement, the algorithm used to solve this problem could be added to the set of requirements based on the results obtained by testing the algorithm on some sample problems.
Table 5.2  Sample Report by Products

COMPANY NAME: ABC CORPORATION
PROJECT MANE: SAMPLE REPORTS
DATE: IO/31/1996
TINE: 09:58

MATERIAL HANDLING REPORT

By Products

Product name: PROD1
200000 IYEAR

<table>
<thead>
<tr>
<th>Production volume:</th>
<th>Distance</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between RECEIVING and ROBOT</td>
<td>550,772Ft.</td>
<td>$1,498</td>
</tr>
<tr>
<td>Between RECEIVING and BROACII</td>
<td>416,244Ft.</td>
<td>$2,008</td>
</tr>
<tr>
<td>Between RECEIVING and MILL</td>
<td>155,739Ft.</td>
<td>$929</td>
</tr>
<tr>
<td>Between ROBOT and MILL</td>
<td>611,200Ft.</td>
<td>$2,380</td>
</tr>
<tr>
<td>Between RECEIVING and ASSEMBLY</td>
<td>290,520 Ft</td>
<td>$2,876</td>
</tr>
<tr>
<td>Between BROACH and ASSEMBLY</td>
<td>173,225 Ft.</td>
<td>$603</td>
</tr>
<tr>
<td>Between MILL and ASSEMBLY</td>
<td>322,562 Ft.</td>
<td>$1,170</td>
</tr>
<tr>
<td>Between ASSEMBLY and SHIPPING</td>
<td>157,6921Ft</td>
<td>$1,465</td>
</tr>
<tr>
<td>Total</td>
<td>2,738,903 Ft</td>
<td>$12,929</td>
</tr>
</tbody>
</table>
Table 5.3 Sample Material Handling Report

COMPANY NAME: ABC CORPORATION
PROJECT NAME: SAMPLE REPORTS
DATE: 10/31/1996
TIME: 09:59

MATERIAL HANDLING REPORT

By workcenters Time unit: YEAR

<table>
<thead>
<tr>
<th>Distance</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between RECEIVING and ROBOT</td>
<td>550112Ft.</td>
</tr>
<tr>
<td>Between RECEIVING and BROACH</td>
<td>705,728Ft.</td>
</tr>
<tr>
<td>Between RECEIVING and MILL</td>
<td>336,163Ft</td>
</tr>
<tr>
<td>Between ROBOT and MILL</td>
<td>672,200Ft</td>
</tr>
<tr>
<td>Between RECEIVING and ASSEMBLY</td>
<td>636,871 Ft.</td>
</tr>
<tr>
<td>Between BROACH and ASSEMBLY</td>
<td>678,563 Ft.</td>
</tr>
<tr>
<td>Between MILL and ASSEMBLY</td>
<td>492,267 Ft.</td>
</tr>
<tr>
<td>Between ASSEMBLY and SHIPPING</td>
<td>807,992 Ft.</td>
</tr>
<tr>
<td>Between RECEIVING and LATHE</td>
<td>334,458 Ft.</td>
</tr>
<tr>
<td>Between LATHE and ASSEMBLY</td>
<td>734,336Ft</td>
</tr>
<tr>
<td>Grand Total</td>
<td>5,948,843 Ft.</td>
</tr>
</tbody>
</table>
CHAPTER 6

THE GROUPING PROBLEM

6.1 Problem Description

The FactoryFlow program in this suite performs material flow analysis for given CAD layouts. The Workcenter Grouping option in FactoryFlow permits product routings to be defined through groups of workcenters. A group defines a collection of workcenters with identical processing capabilities, so the product could be routed to any workcenter in the group. The Group Calc command within the Workcenter Grouping option automatically identifies the specific workcenters to route the products through, as shown in Figure 6.1. The objective is to minimize the total flow cost of all product routings, based on the distances between workcenters. Recent applications have included semiconductor manufacturers planning printed circuit board manufacturing and automotive manufacturers locating receiving docks along the facility perimeter.

The current algorithm implemented in FactoryFlow Version 4.6 has four significant shortcomings. First, the algorithm can handle routings with at most four consecutive groups. Several proposed implementations of the algorithm cannot currently be implemented because of this constraint. The current algorithm generates and stores several possible complete routings through workcenters for analysis. Since there are a combinatorial number of possible routings, the current approach is slow to execute, requiring over 5 hours to solve a large problem. The approach often fails to generate the optimal routing, resulting in increased flow costs.
In the grouping problem, there are parts to be produced via routings through a series of workcenters. Operations performed in groups can be produced on any one of the identical workcenters in the group. Each possible path has a flow cost associated with it, based on the handling devices, unit loads, and distances between workcenters. The objective is to assign a routing path (specific workcenter assignments in each group) to each part to minimize the flow costs over all parts. Each workcenter in a group has a capacity constraint, limiting the amount of product flowing through the workcenter based on setup times, cycle times, and production quantities. If capacity at a workcenter is exceeded, a fraction of a part's production may be assigned to a different path.

An algorithm named PRIORITY was developed by the department of Industrial and Manufacturing Systems Engineering at Ohio University which overcomes the above mentioned limitations. This algorithm is explained in the next section.
6.2 Priority Algorithm Description

Two preliminary algorithms were coded in C on a Unix workstation. Both algorithms created networks of distances between adjacent pairs of workcenters, rather than explicitly creating all possible paths. This shift in logic from the current Factory Flow algorithm will permit better solutions to problems in less time. The priority algorithm which was implemented is shown in Figure 6.2. The distances in the network are distance between workcenters divided by unit load size or distance per unit of product. A shortest path algorithm [13] is used to find the shortest distance path through all groups. The algorithms differed only in the order they select paths to schedule. The Greedy algorithm selected the path with the maximum shortest path distance first. Therefore, it schedules the highest flow cost paths, the paths with more groups per path and smaller unit loads, first. The Priority algorithm estimated the increase in cost if a path is scheduled last instead of first. It scheduled paths with the maximum increase in cost first. If there are no cost increases, the algorithm scheduled the path with the maximum shortest path length first, the same as the Greedy algorithm.

Based on preliminary test results the Priority algorithm had on average 13.0% less move distance than FactoryFlow and 1.6% less move distance than Greedy algorithm. Both new algorithms showed the most improvement on the more difficult problems, where the order paths are scheduled becomes more critical. The Priority algorithm showed slight but consistent improvement over the Greedy algorithm. The Priority algorithm took about twice as long as the Greedy algorithm on all test problems.
However, overall times were short by 0.01 to 0.2 minutes on small and medium problems and 10 minutes on a large problem.

Figure 6.2 The priority algorithm
CHAPTER 7
DISCUSSION OF RESULTS

The software requirements set has been developed based on the facilities design methodology discussed earlier. This requirements set was then used to evaluate the Cimtech software suite.

After evaluating the Cimtech software suite, some of the deficiencies in the suite were identified. The major deficiencies are summarized below:

1. Does not provide a process planning tool.
2. Does not have the provision to design cellular manufacturing systems.
3. Planner cannot design and analyze multifloor layouts.
4. Does not apply the quadratic assignment algorithm for analysis when the departments in the layout have equal areas.

The Cimtech software suite also satisfies most of the requirements stated in the requirements set developed earlier in the thesis. The priority algorithm was then implemented to solve the workcenter grouping problem described earlier.

The priority algorithm was tested on a total of ten problems. Four problems were supplied by Cimtechnologies (listed with an * in the description below). Six modifications were made to the problems, creating a total of ten test problems. Small problems had no more than 3 groups, 5 paths, and 30 workcenters. Medium problems had no more than 9 groups, 60 paths, and 70 workcenters. The large problem had exactly 7 groups, 200 paths, and 550 workcenters.
To test the robustness of the algorithms over a variety of problem types, each problem was designed with unique characteristics, as summarized below. Sample input files are listed in Appendix 3.

**TEST PROBLEM DESCRIPTIONS**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping *</td>
<td>Small electronics problem, multiple identical paths, many bottlenecks.</td>
</tr>
<tr>
<td>Groupina</td>
<td>Medium electronics problem with up to 40 machines per group</td>
</tr>
<tr>
<td>Grp1*</td>
<td>Small size problem with similar paths</td>
</tr>
<tr>
<td>Grp1a</td>
<td>Small problem with a variety of paths</td>
</tr>
<tr>
<td>Grp1b</td>
<td>Small problem with variable length paths</td>
</tr>
<tr>
<td>Grp2 *</td>
<td>Medium size problem</td>
</tr>
<tr>
<td>Grp2a</td>
<td>Medium problem with up to 5 groups per path</td>
</tr>
<tr>
<td>Grp2b</td>
<td>Medium problem with up to 7 groups per path</td>
</tr>
<tr>
<td>Grp2c</td>
<td>Medium problem with many paths to the same groups</td>
</tr>
<tr>
<td>Ohio *</td>
<td>Large problem</td>
</tr>
</tbody>
</table>

The properties code and results obtained on implementation of the priority algorithm with the above mentioned input files are discussed below:

1. Priority is coded as a stand-alone program, PRIORITY.EXE.
2. Priority is coded in C. The personal computer program was compiled using the Borland C++ compiler version 3.1 with the code generation option setting set to model = "large". The program was also compiled using the Unix Gnu compiler version 2.6. The program is listed in Appendix 1.

3. The program reads in data from the six ASCII files. However, no error checking is done regarding files or data. The program assumes that all files have the same name and standard extensions. The program does not check to see if the file exists before trying to open it. The program assumes that all data are present and in the proper format. For example, some of the test files from Cimtechnologies were created outside of FactoryFlow and did not run until missing file headers and data were added. Error checking routines and dialog boxes from the current Group Ca/c program should be added to the Priority algorithm file reading routines.

4. The Priority algorithm has no fixed restrictions on the number of consecutive groups that can be visited. It can handle up to 10,000 distances in its shortest path routine, the equivalent of visiting 25 consecutive groups with 20 machines per group. It was able to handle all the test problems. The algorithm can handle varying product quantities for the same part in its flow cost calculations.

6. The algorithm can handle varying material handling efficiencies in its flow cost calculations.

7. The algorithm handles unique distances between workcenters based on the product and material handling device, as read in from the path file.
8. Flow costs can be calculated based on "move distance" or "shortest path" according to the algorithm movedist variable setting. Cimtechnologies can set this variable according to user dialog box input.

9. The algorithm finds a near optimal solution. The shortest path algorithm implicitly analyzes all possible routings to insure that each part is routed to the optimal available (capacity remaining) workcenters. The global optimum depends on the sequence the parts are routed. The algorithm intelligently sequences parts to minimize the effect of capacity constraints.

10. The algorithm sends the solution output to the MWCP.PRT file. For testing purposes only, the final flow cost is calculated and can be printed to the screen. Sample output file is listed in Appendix 3.

11. The program is on average 90.5% faster than the current algorithm and generated better flow costs as shown in Tables 7.1 and 7.2.

12. The program is completely self-documented, and is listed in the Appendix 2. The Priority algorithm solves the FactoryFlow grouping problem. It generated lower cost solutions, took less time, did not have the same memory errors, and can handle larger routings than the existing FactoryFlow algorithm.
Table 7.1 Flow Cost Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Factory Flow (Feet)</th>
<th>Priority (Feet)</th>
<th>Priority vs. Factory Flow % improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping</td>
<td>14729770</td>
<td>14399858</td>
<td>2.2%</td>
</tr>
<tr>
<td>Groupina</td>
<td>14921024</td>
<td>14548072</td>
<td>2.5%</td>
</tr>
<tr>
<td>Grp1</td>
<td>15441658</td>
<td>15443456</td>
<td>0.0%</td>
</tr>
<tr>
<td>Grp1a</td>
<td>40557372</td>
<td>35657740</td>
<td>12.1%</td>
</tr>
<tr>
<td>Grp1b</td>
<td>43411301</td>
<td>37259528</td>
<td>14.2%</td>
</tr>
<tr>
<td>Grp2</td>
<td>46396982</td>
<td>40813272</td>
<td>12.0%</td>
</tr>
<tr>
<td>Grp2a</td>
<td>------------</td>
<td>39379524</td>
<td>------------</td>
</tr>
<tr>
<td>Grp2b</td>
<td>------------</td>
<td>38908212</td>
<td>------------</td>
</tr>
<tr>
<td>Grp2c</td>
<td>36113513</td>
<td>35181076</td>
<td>2.6%</td>
</tr>
<tr>
<td>Ohio</td>
<td>1803475162</td>
<td>500422776</td>
<td>72.2%</td>
</tr>
<tr>
<td>Average</td>
<td>------------</td>
<td></td>
<td>14%</td>
</tr>
</tbody>
</table>
Table 7.2 Comparison of Execution Times

<table>
<thead>
<tr>
<th>Problem</th>
<th>Factory Flow (minutes)</th>
<th>Priority (minutes)</th>
<th>Priority vs. Factory Flow % Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping</td>
<td>0.15</td>
<td>0.01</td>
<td>93.3 %</td>
</tr>
<tr>
<td>Groupina</td>
<td>1.15</td>
<td>0.16</td>
<td>86.1%</td>
</tr>
<tr>
<td>Grp1</td>
<td>0.13</td>
<td>0.01</td>
<td>92.3%</td>
</tr>
<tr>
<td>Grp1a</td>
<td>0.14</td>
<td>0.01</td>
<td>92.9%</td>
</tr>
<tr>
<td>Grp1b</td>
<td>0.13</td>
<td>0.01</td>
<td>92.3 %</td>
</tr>
<tr>
<td>Grp2</td>
<td>0.60</td>
<td>0.04</td>
<td>93.3 %</td>
</tr>
<tr>
<td>Grp2a</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grp2b</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grp2c</td>
<td>0.42</td>
<td>0.02</td>
<td>95.2%</td>
</tr>
<tr>
<td>Ohio</td>
<td>300</td>
<td>65</td>
<td>72.2</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>90.5 %</td>
</tr>
</tbody>
</table>
CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

This research will give facilities planners better understanding of the important issues to consider in analyzing a facilities planning software. The users of the Cimtech software suite will now be able to judge the scope of the tool. This work also provides knowledge needed in the field of facilities planning to effectively use the software and to properly analyze and judge the accuracy of the results provided by the software and has also discussed an improvement developed for the Cimtech software which was implemented. Integration of the priority algorithm has improved the scope and practical applicability of the Factory Programs software suite. Thus the Priority algorithm is an additional requirement to the already developed software requirements set. The user can now group any number of workcenters together.

The next steps in this research effort should be to:

1. Work towards developing an algorithm which would help design cellular manufacturing systems for the factory programs software.
2. Work towards providing the factory programs suite the ability to design and analyze multi floor layouts.
4. Evaluate other facility design software packages in the market.
CHAPTER 9

REFERENCES


APPENDIX 1

Priority.c
 /******************************************************************************
 *  *
 * Title: Machine grouping algorithm
 *  *
 * Programmers: Shivaram Kota, Thomas Lacksonen
 *  *
 * Ohio University, Dept. of Industrial & Mfg. Systems Engr.
 *  *
 * Date: 9/30/1997
 *  *
 * Version 1.0
 *  *
 * Description: Reads in FactoryFlow data files and
 *  *
 * generates efficient routings by machine in MWCP.PRT
 *  *
 ******************************************************************************/

 #include<stdio.h>
 #include<stdlib.h>
 #include<string.h>
 #include<math.h>

 /* define global problem size limit parameters*/
#define MAXDIST 10000000 /* Maximum distance between any two machines*/
#define MAXPROD 20 /* Maximum number of products in from a product file*/
#define MAXMACH 600 /* Maximum number of machines in the part and group files*/
#define MAXGRP 20 /* Maximum number of groups in the group file*/
#define MAXPRT 600 /* Maximum number of part routings in the parts file*/
#define MAXARC 10000 /* Maximum number of arcs connecting two successsive machines*/
#define MAXNODE 200 /* Maximum number nodes (possible machines) in one path*/
#define MAXMH 20 /* Maximum number of material handling devices in the material handling file*/
#define MAXPATH 26000    /* Maximum number of distances in the path file*/

/* Define function prototypes*/
int read_prodfile(char *, float [], char *[]);
void read_grpfile(char *, char *[], char *[], float *, float *
, int *, int [], int []);
void read_pthfile(char *, char *[], int *, int *, int *,
int *, int [], int [], int, char *[]);
int read_prtfile(char *, int [], char *[], char *[], char *
[], int[], float [], float [], float[], float [],
float [], float [], float [], int [], char *[], int [],
char *[], int, int *, int *, int [], int []);
int is_grp(char *, char *[], int *);
float shortest_path(int, int *, float *, int *, int []);
void priority(char *[], int, char *[], int, int [], int,
char *[], char *[], int *, int *, int, char *[], int [],
float [], float [], float [], float [], float [],
float [], float [], float [], int [], char *[], int [],
char *[], int, int *, int *, int [], int []);
int read_mhfile(char *, float [], char *[]);

/* variables defined in main*/
int distindexl[MAXPROD], distindex2[MAXPROD];    /* first and last distances in path file for a product*/
int *frommachine;     /* number of from machine in path file*/
int *tomachine;       /* number of to machine in path file*/
char *ptfname;        /* path file name with .pth extension*/
int *mach_dist;       /* Distance between two machines in the path file*/
char *grpname[MAXGRP]; /* Group name following the expression ! in the group file*/
char *macname[MAXMACH];/* Names of the machines (workcenters) in group file and parts file*/
char *fname;          /* Group file name with .grp extension*/
float *macnum;        /* Number machines in a workcenter in the group file*/
float *time;          /* Machine time available in the group file*/
int no_ofgrp; /* Number of groups in the group file*/
int no_ofmac; /* Number of machines in the group and part files */
int grpindex[MAXGRP]; /* First machine number of each of the groups */
int i, j, g;
int *path; /* path, or routing, number for each routing in the parts file*/
char *prodname[MAXPROD]; /* Product name in the product file*/
float prodqty[MAXPROD]; /* quantity of product produced per time unit in the product file*/
int noprods; /* Number of products read from the product file*/
char buf[20], buf1[20];
char *filename; /* Product file name with .prd extension*/
char *pfname; /* Path file name with .pth extension*/
char *prtnum[MAXPRT]; /* Part name in the part file*/
char *fwctr[MAXPRT]; /* From workcenter name in the part file*/
char *twctr[MAXPRT]; /* To workcenter name in the part file*/
char *mhand[MAXPRT]; /* material handling device name in the part file*/
float prtprodqty[MAXPRT]; /* Quantity of parts per product in the part file*/
float unitld[MAXPRT]; /* unit load quantity handled in the part file*/
float stptime[MAXPRT]; /* setup time in the part file*/
float lotsize[MAXPRT]; /* lot size per setup in the part file*/
float cyctime[MAXPRT]; /* production cycle time in the part file*/
float scrprate[MAXPRT]; /* scrap rate in the part file*/
int grpcode[MAXPRT]; /* Group code in the part file*/
int noprts; /* Number of parts (lines of routings) in the part file*/
char *filenamel; /* Part file name with .prt extension*/
int nomh;        /* Number of handling devices in
the material handling file*/
char *mhfname;   /* Material hand file name with.mhe
extension*/
char *mname[MAXMH];  /* Material handling device name in
the material handling file*/
float eff[MAXMH];  /* handling device efficiency in
the material handling file*/
char *flname;     /* File name (without extension),
currently entered at the prompt*/
int prodn[MAPRT]; /* First line number of each path
in the part file*/
int mhno[MAPRT];  /* material handling device number
in the part file*/

/* variables defined in priority*/
float dist2[MAPRT];  /* Shortest path distance for a
path*/
float time2[MAXMACH]; /* Time available for machines
after bottlenecks are removed*/
int *pathmach2[MAPRT]; /* Machine numbers of shortest path
for a path*/
int path12[MAPRT], path22[MAPRT]; /* first and last lines
in part file for a path*/
int *machnode;        /* machine number for a given
shortest path node*/
int *index;           /* First arc number in network from a
given node (called funcIndex in shortest_path)*/
int ii[MAXNODE];      /* preceding node to given node in
network (called funcI in shortest_path)*/
int pathmach[MAXNODE]; /* machine numbers of shortest path
selected for adding to routings*/
int temp[MAXNODE];    /* node number on shortest path*/
float *arcd;          /* Length of an arc in network in
inches/unitload (called funcD in shortest_path)*/

/* variables used in shortest path*/
float funcDIST[MAXNODE]; /* Minimum distance from root to a
given node in network*/

...........................
void main(void)      /* start of main*/

/* Allocating memory dynamically to all arrays*/
for(i=0;i<MAXPRT;i++)
{
prtype[i] = (char *)malloc(sizeof(char)*30);
fwctr[i] = (char *)malloc(sizeof(char)*20);
twctr[i] = (char *)malloc(sizeof(char)*20);
}
mhfname=(char *)malloc(sizeof(char) *40);
filename = (char *) malloc(sizeof(char) * 40);
filename1 = (char *) malloc(sizeof(char) * 40);
ptfname = (char *)malloc(sizeof(char) * 40);
mach_dist = (int*)malloc(sizeof(int)*MAXPATH);
frommachine = (int*)malloc(sizeof(int)*MAXPATH);
tomachine = (int*)malloc(sizeof(int)*MAXPATH);
macnum = (float*)malloc(sizeof(float)*MAXMACH);
time = (float*)malloc(sizeof(float)*MAXMACH);
path = (int*)malloc(sizeof(int)*MAXPRT);
for(i=0;i<MAXGRP;i++)
grpname[i] = (char *)malloc(sizeof(char) * 20);
for(i=0;i<MAXMACH;i++)
{
amcname[i] = (char *)malloc(sizeof(char) * 20);
}
for(i=0;i<MAXPROD;i++)
prodname[i] = (char *)malloc(sizeof(char)*30);
for(i=0;i<MAXMH;i++)
mhname[i]=(char*)malloc(sizeof(float) *40);

/* TO BE CHANGED BY CIMTECHNOLOGIES */
printf("enter the file name\n"); /*File name from which the data is to be read.*/
scanf("%s",flname);
/* Add the respective extensions to the file name typed*/
strcpy(frtname,flname);
strcat(frtname,".grp");
strcpy(filename,flname);
strcat(filename,".prd");
strcpy(ptfname,flname);
strcat(ptfname,".pth");
strcpy(filename1,flname);
strcat(filename1, ".prt");
strcpy(mhfname, filename);
strcat(mhfname, ".mhe");

/* Function call to read the group file*/
read_grpfile(fname, grpname, macname, macnum, time,
&no_ofgrp, &no_ofmac, grpindex);

/* Function call to read the product file*/
noprods = read_prodfile(filename, prodqty, prodnname);

/* Function call to read the material handling file*/
nomh=read_mhfile(mhfname, eff, mhname);

/* Function call to read the path file*/
read_pthfile(ptfname, macname, &no_ofmac, mach_dist,
frommachine, tomachine, distindex1, distindex2, noprods,
prodnname);

/* Function call to read the part file*/
noprts = read prtfile(filename1, prodno, prtname,
prtprodqty, fwctr, twctr, mhnno, unitld, stptime, lotsize,
cyctime, scrprate, grpcode, grpname, &no_ofgrp, path,
prodnname, noprts, mhnno, nomh);

/*Function call to the priority algorithm - determine
optimal routings and save to MWCP.PRT file*/
priority(grpname, no_ofgrp, macname, no_ofmac, grpindex,
noprts, fwctr, twctr, path, mach_dist, noprods, prodnname,
prodnno, time, cyctime, stptime, lotsize, scrprate, prodqty,
macnum, prtprodqty, unitld, grpcode, prtname, mhnno, mhname,
nomh, frommachine, tomachine, distindex1, distindex2);
} /*End of main*/

**************************************************************************
*********
* This function reads the product file and stores the data in arrays *
* indexed by product number. *
* Inputs: filename
*
* Outputs: pgty, pname
* Return: the number of products in the file
*
******************************************************************************************
***/

int read_read_product_file(char *filename, float pgty[], char *pname[30])
{
    FILE *fp;
    char c[1000], color[20], timeunit[6]; /* Data read in but not required or stored */
    int lines=0, cal, i=0;
    float konst;                /* Conversion factor to set time units to years */

    /* Open file, skip header lines, reading and storing only the time unit constant */
    if((fp=fopen(filename,"r"))==NULL)
        exit(0);
    while(lines<2)
    {
        fgets(c,1000,fp);
        lines++;
    }
    fscanf(fp,"%s",timeunit);
    lines++;
    while(lines<7)
    {
        fgets(c,1000,fp);
        lines++;
    }
    konst=1;
    if(strcmp(timeunit,"MONTH")==0)
        konst=11.42587;
    else if(strcmp(timeunit,"WEEK")==0)
        konst=48;
    else if(strcmp(timeunit,"DAY")==0)
        konst=240;
    else if(strcmp(timeunit,"SHIFT")==0)
        konst=240;
/ * Read and store data for product i */
fscanf(fp,"%s%f%s%d",pname[i],&pqty[i],color,&cal);
pqty[i]=konst*pqty[i];
i=1;
while(!feof(fp))
{
    if
       ((fscanf(fp,"%s%f%s%d",pname[i],&pqty[i],color,&cal))==4)
           { 
               pqty[i]=pqty[i]*konst;
               i++;
           }
   /*End of while loop*/
fclose(fp);
return i;    /* Return the number of products */
} /* End of function read_prodfile*/

**************************************************************************
**
* This function reads the material handling file and stores the *
* data in arrays by material handling device number. *
* Inputs: mhfname *
* Outputs: eff, mhname *
* Return: the number of material handling devices read in *
**************************************************************************

int read_mhfile(char *mhfname,float eff[MAXMH],char *
mhname[40])
{
    char c[1000];
    int lines=0, i=0;
    float c1, c2, c3, c4, c5, c6, c7;
    FILE *fp;
/* Open file, skip header lines */
if ((fp=fopen(mhfname,"r"))==NULL)
    printf("file does not exist");
while(lines<6)
{
    fgets(c,1000,fp);
    lines++;
}

/* Read in line, saving only name and efficiency columns */
fscanf(fp,"%s%s%f%f%f%f%f%f",mhname[i],c,&c1,&c2,&c3,&c4,&c5,&c6,&c7,&eff[i]);
i=1;
while(!feof(fp))
{
    if
        ((fscanf(fp,"%s%s%f%f%f%f%f%f",mhname[i],c,&c1,&c2,&c3,
        &c4,&c5,&c6,&c7,&eff[i]))==10)
            i++;
}
fclose(fp);
return i; /* Return number of handling devices */
} /*End of the function read_mhfile*/

*******************************************************************************
**********
* This function reads the group file and stores group and
* machine data *
* in arrays. Group names start with the '!' character. *
*
* Inputs: fname *
*
* Outputs: grpname, macname, mnum, mtime, ngrp, nmac, 
grpindex       *
*******************************************************************************
**********

void read_grpfile(char *fname, char *grpname[20], char 
*macname[20], float *mnum, float *mtime, int *ngrp, int 
*nmac, int grpindex[])
{
int i, g=0, j=0; /* j is current machine number, g is current group number */
FILE *fp;

/* Open file and read in each line, assuming it is machine data */
if((fp = fopen(fname,"r"))==NULL)
   exit(0);
 fscanf(fp,"%s%f%f",macname[j], &mnum[j], &mtime[j]);
 while(!feof(fp))
 {  
    if(macname[j][0] == '!!') /* If line is group data, store new group data */
    {
       for(i=1;i<strlen(macname[j]);i++)
          grpname[g][i-1] = macname[j][i];
       grpname[g][i-1] = '\0';
       grpindex[g] = j;
       j = j-1;
       g++;
    }
    j++;
    fscanf(fp,"%s%f%f",macname[j],(mnum + j),(mtime + j));
 }

/* Return number of machines and groups */
ngrp = g;
nmac = j;
grpindex[g]=j;
} /* End of function read_grpfile*/

/************************************************************
****
* This function reads the part file and stores data in arrays *
* by part number.
*
* Product, from workcenters, to workcenters, and handling *
* devices are stored as numbers. Path define the routings. *
*
* Inputs: filnamel, grpname, ngrp, prodname, noprods, mhname, *
  * nomh *
  * Outputs: prodno, prtname, ppqty, fwctr, twctr, mnho, *
  * stptime, lotsize, cyctime, scrprate, grpcode, path *
  * Return: Number of routing lines (parts) in the file *

***********************************************************************
*****/

int read_prtfile(char *filnamel, int prodno[], char *
prtname[30], float ppqty[], char *fwctr[20], char *
twctr[20], int mnho[MAXPRT], float unitld[], float *
stptime[], float lotsize[], float cyctime[], float *
scrprate[], int grpcode[], char *grpname[20], int *ngrp, int *
path, char *prodname[30], int noprods, char *
mhname[40], int nomh)
{
  FILE *fp;
  int lines=0, i=0, test, no_pths = 0; /* i is current line
  number, path is current path number */
  char c[1000], pfprodname[30], mhand[20];

  /* Open file and skip past header lines */
  if((fp=fopen(filnamel,"r"))==NULL)
    exit(0);
  while(lines<=3)
  {
    fgets(c,1000,fp);
    lines++;
  }

  /* first line, including possible extra text. Store data
  by part */
  fscanf(fp,"%s%s%f%s%s%f%f%f%d",pfprodname,
 prtname[i], &ppqty[i], fwctr[i], twctr[i], mhand,
  &unitld[i], &stptime[i], &lotsize[i], &cyctime[i],
  &scrprate[i], &grpcode[i]);
  fgets(c,1000,fp);
/* Convert product name to number. Assumes that a match exists. */
for(test=0; test<no_prods; test++)
{
    if(strcmp(prodname[test], pfprodname) == 0)
        break;
}
prodno[i] = test;

/* Convert handling device name to number. Assumes that a match exists. */
for(test=0; test<no_prods; test++)
{
    if(strcmp(mhname[test], mhand) == 0)
        break;
}
mhno[i] = test;

/* Assign path number to routing (path = 0 means routing is between two group names) */
if(is_grp(fwctr[i], grpname, ngrp) || is_grp(twctr[i], grpname, ngrp))
{
    no_pths++;
    path[i] = no_pths;
}
else
    path[i] = 0;

/* Read data, including possible extra text, line by line. Store data by part */
while(!feof(fp))
{
    i++;
    fscanf(fp, "%s%s%f%s%s%f%f%f%f\d", pfprodname, prtname[i], &ppqty[i], fwctr[i], twctr[i], mhand,
            &unitld[i], &stptime[i], &lotsize[i], &cyctime[i], &scrprate[i], grpcode);
    fgets(c, 1000, fp);

    /* Convert product name to number. Assumes that a match exists. */
    for(test=0; test<no_prods; test++)
if (strcmp (prodname[test], pfprodname) == 0)
    break;
}
prodno[i] = test;

/* Convert handling device name to number. Assumes that a match exists. */
for (test = 0; test < noprods; test++)
{
    if (strcmp (mhname[test], mhand) == 0)
        break;
}
mhno[i] = test;

/* Assign path=0, start new path number, or continue current path number, depending on the routing */
if (!is_grp(fwctr[i], grpname, ngrp) && !is_grp(twctr[i], grpname, ngrp))
    {path[i] = 0;}
else if (is_grp(fwctr[i], grpname, ngrp) &&
    (strcmp(fwctr[i], twctr[i-1]) == 0))
    {path[i] = no_pths;}
else
{
    no_pths++;
    path[i] = no_pths;
}
fclose(fp);
no_pths++; return i; /* Return the number of parts */
} /*End of subroutine read_prtfile*/

/*--------------------------------------------------------------------------
* Called by read_prtfile. It checks if a given workcenter is a group.*
* Inputs: wctr, grpname, ngrps
* Return: a value 1 if the workcenter is a group name,
* else 0             */
int is_grp(char *wctr, char *grpname[20], int *ngrps)
{
    int test = 0;
    int i;
    for(i = 0; i<*ngrps; i++)
    {
        if(strcmp(wctr, grpname[i])==0)
        {
            test = 1;
            break;
        }
    }
    return test;
} /* End of function is_grp*/

// ******************************************************
//       This function reads the path file and stores data in arrays
//       by distance element number.
//       From workcenter and to workcenter are stored as numbers.
//       Inputs: pfname, machname, no_ofmac, prodname, noprods
//       Outputs: mach_dist, frommachine, tomachine, distindex1,
//       Distindex2
// ******************************************************

void read_pthfile(char *pfname, char *machname[20], int *no_ofmac, int *mach_dist, int *frommachine, int *tomachine, int distindex1[MAXPROD], int distindex2[MAXPROD], int noprods, char *pname[30])
{
float oldml, oldm2, dist;
int oldprod=0, el, numMach; /* el is current distance
element number, old prod is current product no. */
char mach1[20], mach2[20], hex[20], name[20], prods[30];
FILE *fp;
numMach = *no_ofmac;
el=0;

/* Open file and read data line at a time */
fp = fopen(pfname, "r");
while (!feof(fp))
{
  fscanf(fp,"%s",name);

  /* PRODUCT is keyword indicating product name for the next
  series of distance elements */
  /* Find product number, i, and update indeces for old and
  current product */
  if (strcmp("PRODUCT:",name)==0)
  {
    fscanf(fp,"%s",prods);
    for(i=0;i<noprods;i++)
      {
        if (strcmp(pname[i],prods)==0)
          break;
      }
    distindex2[oldprod]=el-1;
    distindex1[i]=el;
    oldprod=i;
  }

  /* PATH is keyword for distance element. */
  /* Convert machine names to numbers and store. Round
distance to nearest inch and store as integer. */
  if (strcmp("-PATH:",name)==0)
  {
    fscanf(fp,"%s%s%s%f",mach1,mach2,hex,&dist);
    oldml = 0;
    for (i = 0; i < numMach; i++)
    {
      if (strcmp(mach1, machname[i])==0)
        {
          frommachine[el] = i;
        }
oldml = 1;
    break;
  }
}
if (oldml==0)
{
    strcpy(machname[numMach], machl);
    frommachine[el] = numMach;
    numMach++;
}
oldm2 = 0;
for (i=0; i<numMach; i++)
{
    if (strcmp(mach2, machname[i]) == 0)
    {
        tomachine[el] = i;
        oldm2 = 1;
        break;
    }
}
if (oldm2==0)
{
    strcpy(machname[numMach], mach2);
    tomachine[el] = numMach;
    numMach++;
}
    mach_dist[el] = (dist);
    el++;
}
distindex2[oldprod]=el-1;
fclose(fp);

/* Add FFEXIT dummy machine name to list of machines */
for (i=0; i<noprods; i++)
    strcpy(machname[numMach], "FFEXIT");
numMach++;
*(no_ofmac) = numMach;
} /*End of read_pthfile subroutine*/
This function takes all data read in from the FactoryFlow files and uses it to calculate the minimum distance routing between machines. A two-stage iterative algorithm is used.*

- Each iteration: Schedule one path between workcenters.

- Stage 1: Find minimum distance for each path and find all bottleneck machines, assuming all machines are available.
- Stage 2: Find minimum distance for each path, assuming all bottleneck machines are not available. Schedule the path with the largest increase in minimum distance.

Use maximum distance path as tiebreaker.

Inputs: All variables in the function argument list

Output: A duplicate part file, MWCP.PRT, containing workcenters in place of groups, with quantities that do not exceed workcenter capacities. No output to main.

```
void priority(char *grpname[MAXGRP], int no_ofgrp, char *macname[MAXMACH], int no_ofmac, int grpindex[], int noprts, char *fwctr[MAXPRT], char *twctr[MAXPRT], int *path, int *mach_dist, int noprods, char *prodbname[MAXPROD], int prodno[MAXPRT], float time[MAXMACH], float cyctime[MAXPRT], float stptime[MAXPRT], float lotsize[MAXPRT], float scrprate[MAXPRT], float prodqty[MAXPROD], float macnum[MAXMACH], float qty[MAXPRT], float unitld[MAXPRT], int grpcode[MAXPRT], char *prtname[MAXPRT], int mhno[MAXPRT], char *mhname[MAXMH], int nomh, int...`
*frommachine, int *tomachine, int distindex1[MAXPROD], int distindex2[MAXPROD])
{
    int toindex1, toindex2; /* range of to workcenter numbers for a given group*/
    int findex1, findex2; /* range of from workcenter numbers for a given group*/
    int i, j, k, m, p, x;
    int arcno; /* current number of paths added to the network*/
    int nodeno; /* current number of nodes added to the network*/
    int *jnode; /* succeeding node number for arc in network (called funcJ in shortest_path)*/
    int node1, node2; /* range of node numbers in network for a given group*/
    int loop; /* stage number in an iteration, 1-first stage, 2-second stage*/
    int path1, path2; /* range of line numbers in parts file for the current path*/
    float Dist; /* min distance from a root to the last node in network, returned from shortest_path*/
    float bestdist; /* current best score (max difference between shortest paths in stages 1 and 2)*/
    float minqty; /* part quantity to schedule on a path, minimum of total quantity and all machqty's*/
    float machqty; /* amount of part quantity which uses up capacity for given workcenter on the path*/
    float bestpath; /* path number with current bestdist*/
    float totaldist=0; /* total move distance in feet per unit load for algorithm (to be minimized)*/
    float totalqty=10; /* sum of quantities of all parts yet to be scheduled*/
    float gtime; /* total time available for all workcenters in a group*/
    float spdist; /* shortest path distance of current best path*/
    float d; /* distance in feet between two workcenters from mach dist array*/
    int movedist=1; /* =1 if distance is measured in feet per unit load, =2 if feet per piece*/
int flag;        /* =0 if no bottleneck workstation in
                stage 1 iteration, =1 if there is a bottleneck*/
FILE *cptr;

/* dynamic memory allocation of local arrays*/
jnode= (int *)malloc(sizeof(int) *MAXARC);
machnode = (int *)malloc(sizeof(int) *MAXNODE);
index = (int *)malloc(sizeof(int) *MAXNODE);
arcd = (float *)malloc(sizeof(float) *MAXARC);
for(i=0;i<MAXPRT;i++)
    pathmach2[i] = (int *)malloc(sizeof(int) *MAXNODE);
/* Open default parts file and enter header lines*/
cptr = fopen("MWCP.PRT", "w");
fprintf(cptr, "product part
    Qty/ from to Matl Unit
    Setup Lot Cycle scrap group \n");
fprintf(cptr,"name name product Workcenter Workcenter Hand Load Time
    Size Time Rate Code \n");
    j = grpindex[no_ofgrp];
while(j<no_ofmac)
{
    time[j] = 10;
    j++;
}
flag=1;

/* Find and schedule all paths between 2 fixed workcenters
(path=0)*/
for(i=0;i<noprts;i++)
{
    if(path[i]==0)
        {
        /* Find workcenter numbers */
            j=grpindex[no_ofgrp];
            while(j<no_ofmac)
            {
                if(strcmp(macname[j],fwctr[i])==0)
                    findex1 = j;
                if(strcmp(macname[j],twctr[i])==0)
                    toindex1 = j;
                j++;
            }

            }
/* Find product number and distance between workcenters */
p=prodno[i];
m=mhno[i];
for (j=distindex1[p];j<=distindex2[p];j++)
{

if(findex1==frommachine[j]&&toindex1==tomachine[j])
    break;
}
if(j<=distindex2[p])
    d=mach_dist[j]/12;
else
    d=0;
if(movedist==1)
    totaldist =
totaldist+prodqty[p]*qty[i]*d/(unitld[i]*(eff[m]/100));
else
    totaldist = totaldist+prodqty[p]*qty[i]*d;
fprintf(cpfr,"%26s %26s %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6d\n", prodname[p],
prtno[i], qty[i], macname[findex1], macname[toindex1],
mnemonic[m], unitld[i], stptime[i], lotsize[i], cycntime[i],
scrprate[i], grpcode[i]);
    qty[i] = 0;
}

/* Main iterative loop repeats until all quantities of all parts have been scheduled */
while(totalqty>.0001)
{
    totalqty = 0;
    for(i=0;i<no_ofmac;i++)
    {
        time2[i]=time[i];
    }

    /* Loop=1, find shortest path and bottleneck stations; loop=2, find shortest path skipping all bottlenecks*/
    for(loop=1;loop<=2;loop++)
    {
        if (loop==2||flag==1)
        {
            
        
    
*/
bestpath=0;
bestdist=-MAXDIST;

/* Loop through all paths in parts file*/
for(i=0;i<oprots;i++)
{

/* First routing in a path which has not been completely
scheduled yet. Start a new network.*/
    if(qty[i]>0.0001 " (i==0 || path[i]!=path[i-1]))
    {
        if(loop==2)
            totalqty = totalqty + qty[i];
        for(j=0;j<no_ofgrp;j++)    /* Find
workcenter numbers for first group in path*/
            if(strcmp(grpname[j],fwctr[i])==0)
                { 
            toindex1 = grpindex[j];
            toindex2 = grpindex[j+1]-1;
            }
    }

    j = grpindex[no_ofgrp];
    while(j<no_ofmac)
    {
        if(strcmp(macname[j],fwctr[i])==0)
            { 
            toindex1=j;
            toindex2=j;
            }
        j++;
    }

    arcno = 0;    /* Create dummy arcs
from node 0 to all workcenters at start of path*/
    index[0]=0;
    nodeno=1;    /* Node 0 is dummy start
node*/
    for(k=0;k<=toindex2-toindex1;k++)
        { 
            jnode[arcno] = nodeno + k;
            arcd[arcno] = 0;
            arcno++; }
if (qty[i] > 0.0001)
{
    for (j = 0; j < no_ofgrp; j++)
    { /* Find workcenter numbers for from group in path*/
        if (strcmp(grpname[j], fwctr[i]) == 0)
        {
            findex1 = grpindex[j];
            findex2 = grpindex[j+1]-1;
        }
    }
    j = grpindex[no_ofgrp];
    while (j < no_ofmac)
    { /* Find workcenter numbers for to group in path*/
        if (strcmp(macname[j], fwctr[i]) == 0)
        {
            tindex1 = j;
            tindex2 = j;
        }
        j++;
    }
    for (k = 0; k < no_ofgrp; k++)
    { /* Find workcenter numbers for to group in path*/
        if (strcmp(grpname[k], twctr[i]) == 0)
        {
            tindex1 = grpindex[k];
            tindex2 = grpindex[k+1]-1;
        }
    }
    k = grpindex[no_ofgrp];
    while (k < no_ofmac)
    { /* Find workcenter numbers for to group in path*/
        if (strcmp(macname[k], twctr[i]) == 0)
        {
            tindex1 = k;
            tindex2 = k;
        }
        k++;
    }
node1 = nodeno+findex2-findex1+1;
m=mhno[i];
p=prodno[i];
for(j=findex1;j<=findex2;j++)
{
    machnode[nodeno] = j; /* Identify the workcenter number for given nodeno*/
    index[nodeno]= arcno;
    nodeno++;
    for(k=0;k<= toindex2-toindex1;k++)
    {
        jnode[arcno]=node1+k; /* Identify to node numbers for given arc*/
        if((loop==1 &&
            time[j]>1)||(loop==2&&time2[j]>1)) /* If workcenter has capacity*/
        {
            for(x=distindex1[p];x<=distindex2[p];x++) /* Get distance from mach_dist array*/
            {
                if(j==frommachine[x]&&toindex1+k==tomachine[x])
                    break;
                }/* End of if j==frommachine[x]&&toindex1+k==tomachine[x]*/
                if(x<=distindex2[p])
                    d=mach_dist[x]/12;
                else
                    d=0;
                if(movedist==1) /* Distance in network depends on what user selects to optimize*/
                    arcd[arcno]=d/(unitld[i]*(eff[m]/100));
                else
                    arcd[arcno]=d;
            } else /* If from workcenter has no capacity, put dummy large distance in network*/
                arcd[arcno]=MAXDIST;
        arcno++;
    } /* End of for k=0; k<= toindex2-toindex1; k++*/
} /* End of for loop*/
} /* Last routing in a path which has not been completely scheduled yet*/
          if(qty[i]>0.0001 && (i== noprts || path[i] != path[i+1]))
                
                for(k = 0;k<no_ofgrp;k++) /* Find workcenter numbers for first group in path*/
                
                if(strcmp(grpname[k],twctr[i])==0)
                
                { findex1 = grpindex[k];
                  findex2 = grpindex[k+1]-1;
                }
                
                k = grpindex[no_ofgrp];
                while(k<no_ofmac)
                
                { if(strcmp(macname[k],twctr[i])==0)
                
                  { findex1 = k;
                    findex2 = k;
                  }
                  k++;
                }
                
                k = nodeno+findex2-findex1+1;
                for(j=findex1;j<=findex2;j++)
                
                { machnode[nodeno] = j; /* Identify the workcenter number for given nodeno*/
                  index[nodeno] = arcno;
                  nodeno++;
                  } /* Create single dummy end node at end of path*/
                jnode[arcno]=k;

          if((loop==1&&time[j]>1)||(loop==2&&time2[j]>1))
          
          arcd[arcno] = 0; /* Create dummy arcs to end node from all workcenters at end of path*/
          else /* If from workcenter has no capacity, put dummy large distance in network*/
          
          arcd[arcno]=MAXDIST;
          arcno++;
index[nodeno]=arcno;

/* Calculate shortest path for just created network*/
Dist = shortest_path(nodeno, index, arcd, jnode, ii);    /* Returns
distance of shortest path*/

/* Objective is to max difference, with max sh. path
distance as tiebreaker: max (Dist-dist2)+.001*dist2 */
if((loop==2&&Dist-0.999*dist2[i]>bestdist))
/* New best objective value found */
{
    bestdist = Dist-0.999*dist2[i];  /* Save
objective value*/
    spdist=dist2[i];                /* Save
actual shortest path distance*/
    for(j=0;j<MAXNODE;j++)               /* Save
workcenter numbers on shortest path*/
    {
        pathmach[j]=pathmach2[i][j];
    }
    path1=path12[i];                     /* Save
path indexes of shortest path*/
    path2=path22[i];
}
if(loop==1)    /* In stage 1, just save
shortest path distance and workcenters for all paths*/
{
    dist2[i]=Dist;
    node1=nodeno;
    k = 0;
    while(node1>0)    /* Work backwards from
end node to node 0 to identify sh. path network*/
    {
        temp[k]=ii[node1];
        node1=temp[k];
        k++;
    }
    k--;    
    for(j=0;j<k;j++)
    {
        node1 = temp[k-j-1];
    }
pathmach2[i][j] = machnode[node1]; /*
Store workcenter numbers of the sh. path nodes*/
}
path2=i;
path1=i-k+2;
path22[i] = i;
pathl2[i] = i-k+2;

/* Find quantity of part to schedule - minimum of part qty
and all machqty*/
node2=-1;  /* Bottleneck workcenter -
one found yet*/
minqty=qty[path1];
for(j=pathl; j<=path2; j++)
{
  node1 = pathmach2[i][j-pathl]; /* Find
time required per workcenter on shortest path*/
p=prodnos[j];
  if(node1<grpindex[no_ofgrp] &&
  (cyctime[j]+stptime[j]/lotsize[j])>0)
  {
    machqty = time[node1] *
    macnum[node1] * (1-(scrprate[j]/100)) /
    ((cyctime[j]+stptime[j]/lotsize[j]) * prodqty[p]);
    if(machqty<minqty) /* A bottleneck
    - capacity exceeded before qty is completed*/
    {
      minqty=machqty;
      node2=node1;
    }
  }
}
if(node2>=0)    /* A bottleneck was
found; set time available to 0 for stage 2*/
time2[node2]=0;
} /* end if qty>.001 */
} /* end for i loop through paths in parts
file*/

/* Schedule the path with the best objective function value
after stage 2 only*/
if(loop==2)
{  
    minqty = qty[path1];
    flag=0;  /* No bottleneck on selected path, so
shortest paths will not change next iteration*/
    for(i=path1;i<=path2;i++)  /* Find time
required per workcenter on shortest path*/
    {
        nodel=pathmach[i-path1];
        p=prodno[i];
        if(nodel < grpindex[no_ofgrp] &&
(cyctime[i]+stptime[i]/lotsize[i])>0)
        {
            machqty = time[nodel]*macnum[nodel]* (1-
(scrprate[i]/100)) / ((cyctime[i]+stptime[i]/lotsize[i]) * prodqty[p]);
            if(machqty < minqty)  /* Bottleneck
workcenter found on shortest path*/
            {
                flag=1;
                minqty = machqty;
            }
        }
    }
    for(i=path1;i<=path2;i++)  /* Print each line
of the routing to the new parts file*/
    {
        nodel=pathmach[i-path1];
        node2=pathmach[i-path1+1];
        p=prodno[i];
        m=mhno[i];
        fprintf(cptr,"%-26s %-26s %-6.3f %-13s %-13s
%-5s %-6.3f %-6.3f %-9.3f %-7.3f %-5.3f %-6d
",
        prodname[p], prtnme[i], minqty, macname[node1],
        macname[node2], mhame[m], unitld[i], stptime[i],
        lotsize[i], cyctime[i], scrprate[i], grpcode[i]);
        qty[i] = qty[i] - minqty;
        if(qty[i] < 0.0001)  /* Avoid roundoff
errors on qty calculation*/
            qty[i] = 0;
        nodel = pathmach[i-path1];  /* Update time
available for each workcenter on path*/
        if(nodel < grpindex[no_ofgrp])
        {  
        }
time[node1] = time[node1] - minqty* (cyctime[i] + stptime[i]/lotsize[i]) * prodqty[p] / ((1-(scrprate[i]/100)) * macnum[node1]);
if(time[node1] < 1) /* Avoid roundoff errors on time calculations*/
    time[node1] = 0;
}
totaldist = totaldist + prodqty[p]*minqty*spdist; /* Update total objective function*/
totalqty = totalqty - minqty; /* Quantity left to schedule*/
} /* End if loop=2 */

/* Determine if a group has run out of total capacity (minutes available)*/
for(i=0;i<no_ofgrp;i++)
{
    gtime = 0;
    for(j = grpindex[i];j<grpindex[i+1];j++)
    {
        if(loop==1)
            gtime = gtime + time2[j];
        else
            gtime = gtime + time[j];
    }
    if(gtime < 1) /* Group is out of capacity*/
    {
        fprintf(cptr,"group %d does not have enough capacity\n",i); /* CIMTECHNOLOGY DIALOG BOX*/
        for(j = grpindex[i];j<grpindex[i+1];j++) /* Set workcenters in group back to full capacity*/
        {
            if(loop==1)
                time2[j] = 115200.00;
            else
                time[j] = 115200.00;
        }
    }
} /* end for loop loop */
} /* end while totalqty loop */
float shortest_path( int nodeno, int* funcIndex, float* funcD, int* funcJ, int funcI[MAXNODE])
{
    int i, j, k;
    float v;

    /* Initialize distances to all nodes to a large number*/
    for(i = 1; i<MAXNODE; i++)
    {
        funcDIST[i] = MAXDIST;
    }
    funcDIST[0]=0;
    for(i=0; i<nodeno; i++) /* Check all arcs leaving node i*/
    {
for(k=funcIndex[i];k<=funcIndex[i+1]-1;k++) /* Check distance to succeeding node j from node i*/
{
    v = funcDIST[i] + funcD[k];
    j = funcJ[k];
    if (v < funcDIST[j]) /* If shorter distance to node j is found, update node j*/
    {
        funcI[j] = i;
        funcDIST[j] = v;
    }
}
return funcDIST[j]; /* Distance to node j (last node)*/
} /*End of function shortest_path*/
APPENDIX 2

Sample input files
<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>QTY/TU</th>
<th>COLOR</th>
<th>CAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUCKS</td>
<td>172500.0</td>
<td>YELLOW</td>
<td>1</td>
</tr>
<tr>
<td>SCOOTERS</td>
<td>172500.0</td>
<td>RED</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Matl. Hand.</td>
<td>Inv. Hour</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>FORK</td>
<td>P.PALLET_TRUCK</td>
<td>1500.0</td>
<td>5.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part</td>
<td>Qty/</td>
<td>Prod., WorkCenter, WorkCenter, Size, Time, Rate, Code</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>TRUCKS NUTS</td>
<td></td>
<td>1.0000 A B FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0000000 0.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>TRUCKS NUTS</td>
<td></td>
<td>1.0000 B SHEAR FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0000000 0.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>TRUCKS NUTS</td>
<td></td>
<td>1.0000 SHEAR MILL FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50.000000 1.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>TRUCKS NUTS</td>
<td></td>
<td>1.0000 MILL LATHE FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50.000000 1.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>TRUCKS NUTS</td>
<td></td>
<td>1.0000 LATHE C FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>100.00000 1.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>TRUCKS NUTS</td>
<td></td>
<td>1.0000 C FFEXIT FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0000000 0.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>SCOOTERS BOLTS</td>
<td></td>
<td>1.0000 C LATHE FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0000000 0.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>SCOOTERS BOLTS</td>
<td></td>
<td>1.0000 LATHE MILL FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>100.00000 1.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>SCOOTERS BOLTS</td>
<td></td>
<td>1.0000 MILL SHEAR FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50.000000 1.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>SCOOTERS BOLTS</td>
<td></td>
<td>1.0000 SHEAR B FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50.000000 1.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>SCOOTERS BOLTS</td>
<td></td>
<td>1.0000 B A FORK 10.0000 0.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0000000 0.0000 0.000 0</td>
<td></td>
</tr>
<tr>
<td>SCOOTERS</td>
<td>BOLTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0000 A</td>
<td>FFEXIT FORK 10.0000 0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.000000 0.0000 0.000 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PATH FILE

PRODUCT: TRUCKS
-PATH: A B 5733 480.00
-PATH: B SHEAR4 5739 300.00
-PATH: B SHEAR3 573F 300.00
-PATH: B SHEAR2 5745 540.00
-PATH: B SHEAR1 5748 540.00
-PATH: SHEAR4 MILL1 5751 1500.00
-PATH: SHEAR4 MILL2 5757 1500.00
-PATH: SHEAR4 MILL3 575D 1380.00
-PATH: SHEAR4 MILL4 5763 1140.00
-PATH: SHEAR3 MILL1 5769 1380.00
-PATH: SHEAR3 MILL2 576F 1260.00
-PATH: SHEAR3 MILL3 5775 1140.00
-PATH: SHEAR3 MILL4 577B 1020.00
-PATH: SHEAR2 MILL1 5781 1260.00
-PATH: SHEAR2 MILL2 5787 1020.00
-PATH: SHEAR2 MILL3 578D 900.00
-PATH: SHEAR2 MILL4 5793 900.00
-PATH: SHEAR1 MILL1 5799 1020.00
-PATH: SHEAR1 MILL2 579F 1020.00
-PATH: SHEAR1 MILL3 57A5 900.00
-PATH: SHEAR1 MILL4 57A6 660.00
-PATH: MILL1 LATHE4 57B1 660.00
-PATH: MILL1 LATHE3 57B7 780.00
-PATH: MILL1 LATHE1 57BD 1020.00
-PATH: MILL1 LATHE2 57C3 1020.00
-PATH: MILL2 LATHE4 57C9 900.00
-PATH: MILL2 LATHE3 57CF 900.00
-PATH: MILL2 LATHE1 57D5 1020.00
-PATH: MILL2 LATHE2 57DB 1020.00
-PATH: MILL3 LATHE4 57E1 1020.00
-PATH: MILL3 LATHE3 57E7 1020.00
-PATH: MILL3 LATHE1 57ED 1020.00
-PATH: MILL3 LATHE2 57F3 900.00
-PATH: MILL4 LATHE4 57F9 1020.00
-PATH: MILL4 LATHE3 57FF 1020.00
-PATH: MILL4 LATHE1 5805 1020.00
-PATH: MILL4 LATHE2 580B 780.00
-PATH: LATHE4 C 5811 1380.00
-PATH: LATHE3 C 5817 1140.00
-PATH: LATHE1 C 581D 900.00
-PATH: LATHE2 C 5823 900.00
PRODUCT: SCOOTERS
-PATH: C LATHE4 5829 1380.00
-PATH: C LATHE3 582F 1140.00
-PATH: C LATHE1 5835 900.00
-PATH: C LATHE2 583B 900.00
-PATH: LATHE4 MILL1 5841 660.00
<table>
<thead>
<tr>
<th>Path</th>
<th>Machine</th>
<th>Code</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATHE4 MILL2</td>
<td></td>
<td>5847</td>
<td>900.00</td>
</tr>
<tr>
<td>LATHE4 MILL3</td>
<td></td>
<td>584D</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE4 MILL4</td>
<td></td>
<td>5853</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE3 MILL1</td>
<td></td>
<td>5859</td>
<td>780.00</td>
</tr>
<tr>
<td>LATHE3 MILL2</td>
<td></td>
<td>585F</td>
<td>900.00</td>
</tr>
<tr>
<td>LATHE3 MILL3</td>
<td></td>
<td>5865</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE3 MILL4</td>
<td></td>
<td>586B</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE1 MILL1</td>
<td></td>
<td>5871</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE1 MILL2</td>
<td></td>
<td>5877</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE1 MILL3</td>
<td></td>
<td>587D</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE1 MILL4</td>
<td></td>
<td>5883</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE2 MILL1</td>
<td></td>
<td>5889</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE2 MILL2</td>
<td></td>
<td>588F</td>
<td>1020.00</td>
</tr>
<tr>
<td>LATHE2 MILL3</td>
<td></td>
<td>5895</td>
<td>900.00</td>
</tr>
<tr>
<td>LATHE2 MILL4</td>
<td></td>
<td>589B</td>
<td>780.00</td>
</tr>
<tr>
<td>MILL1 SHEAR4</td>
<td></td>
<td>58A1</td>
<td>1500.00</td>
</tr>
<tr>
<td>MILL1 SHEAR3</td>
<td></td>
<td>58A7</td>
<td>1380.00</td>
</tr>
<tr>
<td>MILL1 SHEAR2</td>
<td></td>
<td>58AD</td>
<td>1260.00</td>
</tr>
<tr>
<td>MILL1 SHEAR1</td>
<td></td>
<td>58B3</td>
<td>1020.00</td>
</tr>
<tr>
<td>MILL2 SHEAR4</td>
<td></td>
<td>58B9</td>
<td>1500.00</td>
</tr>
<tr>
<td>MILL2 SHEAR3</td>
<td></td>
<td>58BF</td>
<td>1260.00</td>
</tr>
<tr>
<td>MILL2 SHEAR2</td>
<td></td>
<td>58C5</td>
<td>1020.00</td>
</tr>
<tr>
<td>MILL2 SHEAR1</td>
<td></td>
<td>58CB</td>
<td>1020.00</td>
</tr>
<tr>
<td>MILL3 SHEAR4</td>
<td></td>
<td>58D1</td>
<td>1380.00</td>
</tr>
<tr>
<td>MILL3 SHEAR3</td>
<td></td>
<td>58D7</td>
<td>1140.00</td>
</tr>
<tr>
<td>MILL3 SHEAR2</td>
<td></td>
<td>58DD</td>
<td>900.00</td>
</tr>
<tr>
<td>MILL3 SHEAR1</td>
<td></td>
<td>58E3</td>
<td>900.00</td>
</tr>
<tr>
<td>MILL4 SHEAR4</td>
<td></td>
<td>58E9</td>
<td>1140.00</td>
</tr>
<tr>
<td>MILL4 SHEAR3</td>
<td></td>
<td>58EF</td>
<td>1020.00</td>
</tr>
<tr>
<td>MILL4 SHEAR2</td>
<td></td>
<td>58F5</td>
<td>900.00</td>
</tr>
<tr>
<td>MILL4 SHEAR1</td>
<td></td>
<td>58FB</td>
<td>660.00</td>
</tr>
<tr>
<td>SHEAR4 B</td>
<td></td>
<td>5901</td>
<td>300.00</td>
</tr>
<tr>
<td>SHEAR3 B</td>
<td></td>
<td>5907</td>
<td>300.00</td>
</tr>
<tr>
<td>SHEAR2 B</td>
<td></td>
<td>590D</td>
<td>540.00</td>
</tr>
<tr>
<td>SHEAR1 B</td>
<td></td>
<td>5913</td>
<td>540.00</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td>5919</td>
</tr>
</tbody>
</table>
APPENDIX 3

Sample output file
### MWCP.PRT

<table>
<thead>
<tr>
<th>product</th>
<th>part</th>
<th>Qty/Lot</th>
<th>Workcenter</th>
<th>Workcenter</th>
<th>Hand</th>
<th>Load</th>
<th>Time</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>to</td>
<td></td>
<td>name</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle</td>
<td>scrap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matl</td>
<td>Unit</td>
<td>Setup</td>
<td>Lot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRUCKS</th>
<th>NUTS</th>
<th>1.000</th>
<th>A</th>
<th>B</th>
<th>FORK</th>
<th>10.000</th>
<th>0.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRUCKS</th>
<th>NUTS</th>
<th>1.000</th>
<th>C</th>
<th>FFEXIT</th>
<th>FORK</th>
<th>10.000</th>
<th>0.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>1.000</th>
<th>B</th>
<th>A</th>
<th>FORK</th>
<th>10.000</th>
<th>0.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>1.000</th>
<th>A</th>
<th>FFEXIT</th>
<th>FORK</th>
<th>10.000</th>
<th>0.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.666</th>
<th>C</th>
<th>LATHE2</th>
<th>FORK</th>
<th>10.000</th>
<th>0.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.666</th>
<th>LATHE2</th>
<th>MILL4</th>
<th>FORK</th>
<th>10.000</th>
<th>0.200</th>
<th>100.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.666</th>
<th>MILL4</th>
<th>SHEAR1</th>
<th>FORK</th>
<th>10.000</th>
<th>0.100</th>
<th>50.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.666</th>
<th>SHEAR1</th>
<th>B</th>
<th>FORK</th>
<th>10.000</th>
<th>0.100</th>
<th>50.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.334</th>
<th>LATHE1</th>
<th>LATHE1</th>
<th>FORK</th>
<th>10.000</th>
<th>0.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.334</th>
<th>LATHE1</th>
<th>MILL3</th>
<th>FORK</th>
<th>10.000</th>
<th>0.200</th>
<th>100.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCOOTERS</th>
<th>BOLTS</th>
<th>0.334</th>
<th>MILL3</th>
<th>SHEAR2</th>
<th>FORK</th>
<th>10.000</th>
<th>0.100</th>
<th>50.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Resource</td>
<td>Quantity</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOOTERS</td>
<td>SHEAR2</td>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td>0.334</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR3</td>
<td>MILL3</td>
<td>1.000</td>
<td>0.000</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>MILL3</td>
<td>LATHE1</td>
<td>1.000</td>
<td>0.000</td>
<td>0.334</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>LATHE1</td>
<td>C</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.200</td>
<td>100.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR3</td>
<td>MILL1</td>
<td>1.000</td>
<td>0.000</td>
<td>0.334</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>MILL1</td>
<td>LATHE3</td>
<td>1.000</td>
<td>0.000</td>
<td>0.334</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>LATHE3</td>
<td>C</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.200</td>
<td>100.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR2</td>
<td>MILL2</td>
<td>1.000</td>
<td>0.000</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>MILL2</td>
<td>LATHE3</td>
<td>1.000</td>
<td>0.000</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>LATHE3</td>
<td>C</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.200</td>
<td>100.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR4</td>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
<th>Resource</th>
<th>Quantity</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOOTERS</td>
<td>SHEAR2</td>
<td>1.000</td>
<td>0.000</td>
<td>0</td>
<td>0.334</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR3</td>
<td>MILL3</td>
<td>1.000</td>
<td>0.000</td>
<td>0.333</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>MILL3</td>
<td>LATHE1</td>
<td>1.000</td>
<td>0.000</td>
<td>0.334</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>LATHE1</td>
<td>C</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.200</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR3</td>
<td>MILL1</td>
<td>1.000</td>
<td>0.000</td>
<td>0.334</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>MILL1</td>
<td>LATHE3</td>
<td>1.000</td>
<td>0.000</td>
<td>0.334</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>LATHE3</td>
<td>C</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.200</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>SHEAR2</td>
<td>MILL2</td>
<td>1.000</td>
<td>0.000</td>
<td>0.333</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>MILL2</td>
<td>LATHE3</td>
<td>1.000</td>
<td>0.000</td>
<td>0.333</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>LATHE3</td>
<td>C</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.200</td>
</tr>
<tr>
<td>TRUCKS</td>
<td>B</td>
<td>NUTS FORK</td>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>
TRUCKS NUTS 0.001
SHEAR4 MILL1 FORK 10.000 0.100 50.000
1.000 0.000 0
TRUCKS NUTS 0.001
MILL1 LATHE4 FORK 10.000 0.100 50.000
1.000 0.000 0
TRUCKS NUTS 0.001
LATHE4 C FORK 10.000 0.200 100.000
1.000 0.000 0
totaldist = 15443456.000000