Genetic Algorithms for Optimizing Manufacturing Facility Layout

Nur Fadhillah Binti Saleh ¹, Abdul Razak Bin Hussain ²
Faculty Information and Communication Technology
Universiti Teknikal Malaysia, Melaka
75450, Ayer Keroh, Melaka
016-2527610¹, 013-6306712²
nurfadhillah_saleh@yahoo.com¹ razak@utem.edu.my²

ABSTRACT
This paper highlights the preliminary background work dealing with the optimization of the manufacturing facility layout design. It discusses genetic algorithms (GA) - its background, applications, operators and relevant processes. This paper also brief about the manufacturing industries - their types and various facility layouts, the various manufacturing facility layout designs and their corresponding algorithms are tabulated. The successful applications of GA in the facilities layout designs motivate to further seek the applicability of GA in the design of the distributed manufacturing layout.

Keywords
Optimization, Genetic Algorithms, Manufacturing Facility Layout.

1. INTRODUCTION
The layout of the manufacturing facility plays a crucial role in determining the manufacturing operational efficiency. Since 1960s, there has been increasing interest in emulating living being capabilities to solve such hard optimization problems. Simulating the natural evolutionary process of human being results in stochastic optimization techniques called evolutionary algorithms, which can often outperform conventional optimization methods when applied to difficult real world problems.[28] Many optimization problems from the industrial engineering world, in particular the manufacturing systems, are very complex in nature and quite hard to solve by conventional optimization techniques. [29][18]

Recently, genetic algorithms have received considerable attention regarding their potential as an optimization technique for complex problems and have been successfully applied in the area of industrial engineering. The well-known application include scheduling and sequencing, group technology, facility layout and location, transportation and many others.[10] The success of genetic algorithms in various fields has not only encouraged many researchers to explore further into genetic algorithms as an optimization technique for complex problems, but also has led to many new genetic algorithms being developed specifically for these problems.

2. GENETIC ALGORITHMS
2.1 Background
The people that develop the concept for genetic algorithms are John Holland, a professor at the University of Michigan, together with his colleagues and students. In 1975 he published a book, “Adaptation in Natural and Artificial Systems”, in which he present the theory behind genetics algorithms and explores practical applications. Holland is considered the father of genetic algorithms.

2.2 Basic Principles
Genetic Algorithms (GA) are stochastic search techniques based on the mechanism of natural selection and genetic.[11] The basic concept of GA is designed to simulate processes in natural system necessary for evolution. As such that represents an intelligent exploitation of a random search within a defined search space to solve a given problem. It is one of the ways that programs can find potential solutions to complex NP-hard problems. A NP-hard problem is the problem where the number of steps required solving the problem increases at very high rate as the number of unit in the program increases.

GA are typically characterized by the following aspects (Goldberg, 1987):
- GA work with the base in the code of the variables group (artificial genetic strings) and not with the variables in themselves.
- GA work with a set of potential solutions (population) instead of trying to improve a single solution.
- GA do not use information obtained directly from the object function, of its derivatives, or of any other auxiliary knowledge of the same one.
- GA apply probabilistic transition rules, not deterministic rules.

2.3 Application of GA
Genetic algorithms are adaptive search algorithms, which can be used for many purposes. GAs’ are based upon the principles of evolution and natural selection. GAs’ are adept at searching large, non-linear search spaces. A non-linear search space refers to such a large number of potential solutions that the optimal solution can not be solved by conventional iterative means. GAs’ are most efficient and appropriate for situations such as the following:
- Search space is large, complex, or not easily understood.
• There is no programmatic method that can be used to narrow the search space
• Traditional optimization methods are not sufficient

Genetic algorithms may be utilized in solving a wide range of problems across multiple fields such as science, business, engineering, and medicine. The following provides a few examples:

• Optimization: job-shop scheduling [25], machine scheduling [14], call routing for call centers, transportation problem [23], determining electrical circuit layouts.
• Machine learning: designing neural networks [8], designing and controlling robots
• Business applications: utilized in financial trading, credit evaluation, budget allocation, fraud detection

Many optimization problems are non-linear in behavior and are too complex for traditional methods. The set of possible solutions for these problems can be enormous. Genetic algorithms possess the ability to search large and complex search spaces to efficiently determine near optimal solutions in reasonable time frames by simulating biological evolution. [38][39]

2.4 Term or Operator of GA

In a genetic algorithm, each potential solution to the problem is denoted by the term chromosome, which represents the encoded values of each of the variables of the problem. At the start of the genetic algorithm process, a collection, or initial generation, of randomly constructed chromosomes (potential solutions) is created. The genetic algorithm then follows a three-step process to transform the current generation of solutions to the next generation of solutions. This process provides a model of the natural world in which the only operators required for transforming solutions from one generation to the next are reproduction, crossover, and mutation.

The reproduction step selects from the current generation of chromosomes those that will survive to the next generation. This selection uses a probabilistic survival of the fittest mechanism based on a problem-specific evaluation of the chromosomes (modeling the concept of "fitness" from natural selection). This probabilistic survival mechanism is what directs the generations, over time, toward the most valuable (fit) chromosomes.

The crossover step then allows the introduction of new chromosomes into the population of potential solutions by randomly combining pairs of randomly selected existing chromosomes. This step simulates the crossover phenomena in biological reproduction where two chromosomes mate by splitting into randomly selected subsections and then rejoining to form two new chromosomes. The key point here is that crossover allows new chromosomes not present in the current generation to be added to the next generation.

Finally, the mutation step allows the random mutation of existing chromosomes so that new chromosomes may contain parts not found in any existing chromosomes. Again, this step simulates the natural phenomena of gene mutation during reproduction. In the genetic algorithm process, mutation allows the introduction of new solution elements by permitting the occasional random alteration of a single element of the chromosome encoding.

This three-step process is repeated, moving from generation to generation, until a generation is found that is "good enough". There are many ways to determine when a generation is good enough, for example by using the "value" of the best chromosome in the population or by simply iterating for a fixed number of generations. When the process terminates, the best chromosome selected from among the final generation is the solution to the problem. The following also presents a summary of this genetic algorithm process.

<table>
<thead>
<tr>
<th>Compute the initial generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>While (more generations) {</td>
</tr>
<tr>
<td>Reproduce the current generation</td>
</tr>
<tr>
<td>Crossover the current generation</td>
</tr>
<tr>
<td>Mutate the current generation</td>
</tr>
<tr>
<td>Increment the next generation]</td>
</tr>
<tr>
<td>Report the “best” solution in the generation</td>
</tr>
</tbody>
</table>

Figure 2.1: Pseudo code of Genetic Process

This genetic algorithm process has been shown to produce near optimal solutions to a wide variety of difficult optimization problems. [1]

2.4.1 Operator Examples

(a) Chromosome

In genetic algorithms, a chromosome (also sometimes called a genome) is a set of parameters which define a proposed solution to the problem that the genetic algorithms are trying to solve. The chromosome is often represented as a simple string; although a wide variety of other data structures are also used. The usual method of applying genetic algorithms to real-parameter problems is to encode each parameter as a bit string using either a standard binary coding or Gray coding. The bit strings for the parameter are concatenated together to give a single bit string or chromosome which represent the entire vector of parameters. In biological terminology, each bit position corresponds to a gene of the chromosome and each bit value corresponds to an allele.

(b) Fitness Function

The fitness function (FF) is one of the key elements of GA as it determines whether a given potential solution will contribute its elements to future generation through the reproduction process. The FF should be able to provide a good measure of the quality of the solution and should differentiate between the performances of different strings. The first step is to choose the individuals which will have a shot to becoming the parents of the next generation.
This is called the selection procedure, and its purpose is to choose those individuals from the current population which will go to intermediate population (IP). To perform the selection, the GA agent will require a fitness function. This will assign the real number to each individual in the current generation. The fitness function will use an evaluation function to calculate a value of worth for the individual so they can be compared against each other.

(c) Crossover

In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next generation. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based.

One-point crossover

A single crossover point on both parents’ chromosome strings is selected. All data beyond that point in either chromosome is swapped between the two parent chromosomes. The resulting chromosomes are the children:

![Figure 2.2: One-Point Crossover](image)

Two-point crossover

Two point crossover calls for two points to be selected on the parent chromosomes strings. Everything between the two points is swapped between the parent chromosomes, rendering the two child chromosomes:

![Figure 2.3: Two-Point Crossover](image)

“Cut and Slice”

Another crossover variant, the cut and slice approach, result in a change in length of the children strings. The reason for this difference is that each parent’s string has a separate choice of crossover point.

(d) Mutation

Mutation is a background operator which produces spontaneous random changes in various chromosomes. A simple way to achieve mutation would be to alter one or more genes. In genetic algorithms, mutation serves the crucial role of either (a) replacing the genes lost from the population during the selection process so they can be tried in a new context or (b) providing the genes that were not present in the initial population.

The mutation rate is defined as the percentage of the total number of genes in population. The mutation rate controls the rate at which new genes are introduced into the population for trial. If it is too low, many genes that would have been useful and never tried out, but if it is too high, there will be much random perturbation, the offspring will start losing their resemblances to the parents, and the algorithm will lose the ability to learn from history of the search. [40]

3. FACILITY MANUFACTURING LAYOUT

3.1 Manufacturing Industries Background

Manufacturing is a very important commercial activity carried out by the companies that sell products to customers. The type of manufacturing is depending on the kind of product made. Manufacturing industries are industries consist of enterprise and organizations that produce or supply goods or services [13].

Industries can be classified as:

- Primary: Cultivate and Exploit natural resources
- Secondary: Convert the output of primary industries into products
- Tertiary: Constitute economy services sectors.
### 3.2 Facility Manufacturing Layout

One of the important factors to consider in designing a manufacturing facility is finding an effective layout. A general definition of plant layout problem is to find the best arrangement of physical facilities to provide an efficient operation. [24]

1. Distributed Layouts: The distributed layout concept is based on notation that disaggregating large functional departments into smaller sub-departments and distributing them throughout the plant floor can be useful strategy in highly versatile environments. The infrastructure that is typically shared by a single consolidated departments in job shops, such as operators, computer control system, loading/unloading areas, and waste disposal facilities, must be duplicated in a distributed layout across all department copies [5].

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Industry</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wisten and Shayan</td>
<td>2007</td>
<td>Furniture Production Line</td>
<td>Formal Method</td>
</tr>
<tr>
<td>Pérez et al. [27]</td>
<td>2004</td>
<td>Milk Goats Livestock</td>
<td>Genetic Algorithms</td>
</tr>
<tr>
<td>Arapoglu et al. [3]</td>
<td>-</td>
<td>Input and Output Station Location</td>
<td>Genetic Algorithms</td>
</tr>
</tbody>
</table>

2. Modular Layouts: This focus of this approach is on design of customized layouts for facilities with multiple products. It based on idea that layouts can be constructed as a network of basic modules. The use of modules is motivated by the fact that none of the prevailing layout configuration (functional, flow line, and cellular) can individually describe the complex material flow network in a multiple-product manufacturing facility [5]. There is two type of modular layout: Cellular. [24][4][17] and Functional layout.

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Industry</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solimanpura et al. [30]</td>
<td>2005</td>
<td>Production</td>
<td>Ant Algorithms</td>
</tr>
<tr>
<td>Irani et al. [20]</td>
<td>-</td>
<td>Multiple Product</td>
<td>Unification of Matrix, String and Graph Representations of Material Flow Networks</td>
</tr>
</tbody>
</table>

3. Reconfigured Layouts: Reconfigured layouts happen when there is a case where resources can be easily moved around so that frequent relocation of department is feasible. This is motivated by the fact of many industries (e.g., consumer electronic, home appliances, garment manufacturing etc), fabrication and assembly workstations are light and easily relocated. [1][5]

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Industry</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael et al. [21]</td>
<td>2002</td>
<td>Construction Site Layout in Project Planning</td>
<td>Genetic Algorithms</td>
</tr>
</tbody>
</table>

4. Agile Layouts: Most existing layout procedures are based on a static measure of material handling cost. However, this measure do not take into account the effect of layout on operational performance such as cycle time, work-in-process (WIP) accumulation, queuing times at departments and throughput rate. As production planning period shrink, these measure become increasingly important to the performance of the agile factory where reducing manufacturing cycle times and keeping low inventory levels is a key to competitiveness. [6][22][5]

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Industry</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stachowiak and Fertsch [31]</td>
<td>-</td>
<td>Market</td>
<td>Genetic Algorithms</td>
</tr>
</tbody>
</table>

5. Different type of layouts, it means it can be use of any type of layout that we want to use it because it not specific in the certain layout.

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Industry</th>
<th>Algorithm</th>
</tr>
</thead>
</table>
4. CONCLUSION & FUTURE WORK

Our initial study shows that GA has been used in designing various manufacturing facility layout. This serves as our motivation to seek the possibility of using GA in the design of distributed manufacturing layout. Our future work include the design and selection the GA class diagram prior of its implementation using Java. Our proposed algorithm will be tested using simulated and actual layouts, in the hope of finding the deal GA parameter for the distributed manufacturing layout.

5. ACKNOWLEDGMENTS

This work is supported by the grant scheme FRGS/2007/FTMK (2)-F0049.

6. REFERENCES


[31] Stachowiak, A and, Fertsch, M., “Evolutionary strategies in agile facility design”, Computing and Management Department,University of Technology, Poznan.


Internet Accesses

[39] http://www.karnig.co.uk/ga/content.html. (Last access date: 31-12-2007)

Since fitness values are frequently recalculated (the diversity of the population decreases as the algorithm runs), a good strategy to improve the performance of a GA is to reduce the time needed to calculate the fitness. Details depend on implementation, but previously calculated fitness values can often be efficiently saved with a hash table. This kind of optimization can drop computation time significantly (e.g. "IMPROVING GENETIC ALGORITHMS PERFORMANCE BY HASHING FITNESS VALUES" - RICHARD J. POVINELLI, XIN FENG reports that the application of hashing to a GA can improve performance.)