# Priority Dispatching Rules for Virtual Manufacturing Using Genetic Algorithm

**Akela M. Al-Atroshi**
*Technical Institution*

**Abdulsatar M. Khudur**
*Technical Institution*

**Sama Talee Azez Al-Aubaidy**
*College of Computer sciences and Mathematics*
*University of Mosul*

Received on:1/7/2007 Accepted on:26/9/2007

## الملخص

يركز البحث الحالي على تصميم نظام معلومات ذكي وتطبيقه باستخدام لغـــة (Oracle) على أساس خلايا تصنيعية وفق مفهوم(Multi- Agents) إذ تختص هذه الخلايا بإنتاج منتوجات جديدة في إطار الإمكانيات المتاحة للمصنع أو من خلال التعاون مع شركات أخرى ضمن مفهوم التصنيع الافتراضي ، وإن كل خلية (Agent) لها أدوار وصلاحيات خاصة بها . و يركز البحث على خلية التخطيط للأسبقيات لتحديد وقت التسليم من خلال اســتخدام الخوارزميـــات الجينيــة لمحاكاة خطوط الإنتاج وتحديد الأسبقيات لإصدار الأوامر استناداً إلى قواعد مخصــصة لتحديــد المهل الزمنية للمنتوج . هذا وتتجلى أهمية البحث في تصميم برمجيـــات بلغـــة ++C لمحاكـــاة عمليات التصنيع وفق تقنية الخوارزمية الجينية ولتحقيق :

1. أفضل تسلسل لتنفيذ الأعمال وفق القواعد المطلوبة
2. تقليل وقت المنتوج والأجزاء الداخلة في تركيبه
3. أفضل استخدام للموارد المتاحة

أثبتت نتائج التصميم والتطبيق أن عمليات التخطيط بواسطة فلسفة الخوارزميات الجينيـــة تؤدي دورا كبيرا في حساب المهل الزمنية الصناعية في عمليات التصنيع . ويدعم هـــذا الـــدور فلسفة التصنيع الافتراضي في الاحتساب السريع للمهل الزمنية. كذلك أكدت نتـــائج التطبيـــق أن كفاءة البرمجيات المصممة تعتمد على عدد الأعمال المتاحة في وقت التنفيذ ، فكلما كـــان عـــدد الأعمال أكبر فإن كفاءة التنفيذ تكون أفضل .

## ABSTRACT

The current research concentrates on designing and applying an intelligent information system by the use of (Oracle) language based Multi-Agents manufacturing process to produce a new product. Every agent (user)

has its own *roles and privileges*. The research focuses on determining the ***delivery date*** through using genetic algorithms to simulate Shop floor and specify priorities for ***dispatching orders*** according to specific rules which determine the lead time of the product. The importance of the research stems from designing software in c++ to simulate manufacturing processes in the genetic algorithm to realize the following :

1. Attain the best sequences in implementing jobs according to the required rules.
2. Decreasing the queuing time for products and their components in the production processes.
3. Perfect utilization of the available resources.

The results of the designed system application have revealed that the operations planning by the use of the GA philosophy will perform a great role in calculating the product's lead time at the manufacturing operations' stages. This role supports the VM philosophy in calculating the industrial part of the products lead time quickly. Also the application results have confirmed that the designed *GA software* efficiency depends upon the *number **of jobs available at the time of execution***; whenever the number of jobs is bigger, the software execution efficiency is better.

## 1. Introduction to the Problem Domain

The worldwide competition and computer revolution have dramatically changed the market characteristics. Today, it is the buyers, market, where capacity is higher than demand. Price is dictated by the market; it is not based on the producers expected profit [14]. Manufacturers must now develop a balance between the internal and external measures. Their focus is changing from local to global optimization by determining *what customers want* and delivering the right products. This focus and advances in technology are decreasing product life cycle, which together with the high cost of capital make the product time to market one of the major business drivers for many industries. [8] [10] .

The operational unit in Agile Manufacturing Industry is a dynamic virtual enterprise consisting of Multi-Agents who collaborate with each other. The overall objective of an Agile Manufacturing Industry is to achieve quick response to marketing demands, with compatible product quality and lower manufacturing cost [11] [17].

The process of determining the ***industrial lead-time*** faces great challenges according to Agile Manufacturing. This could be attributed to the great individual and variable demands of the customers. The ***problem of the research*** is about the necessity to count the industrial lead-time for each product accurately and precisely when applying the concept of virtual manufacturing. The constant dialogue between the customer and specialists in scheduling manufacturing processes ends when the product cost and its exact delivery date (which constitute the greatest part of the lead time ) are determined .

The current research focuses on determining the ***delivery date*** through using genetic algorithms to simulate Shop floor and specify priorities for dispatching orders according to specific rules which determine the lead time of the product. The ***importance of the research*** stems from designing software to stimulate manufacturing processes in the genetic algorithm to realize the following :

1. Attain the best sequences in implementing jobs according to the required rules.
2. Decreasing the lead-time for products and their components .
3. Perfect utilization of the available resources.
4. Realizing integration between the planning and excessive aspects of jobs.

In order to ***framework this research***, the following ***hypothesis*** has been adopted: a rapid response to individual and variable demands of the customers according to the philosophy of the *virtual manufacturing* requires adopting one of the techniques of Artificial intelligence for determining the sequence of jobs in the production lines in order to exactly determine the lead time.

The ***basic objectives*** of the research are embodied in designing software using C++ language and genetic algorithm to plan priorities for issuing job commands in order to identify the lead time for each product (in priority planning agent). This Software is characterized by its compatibility with the other agents and within Oracle framework and the proposed system in the thesis. All that is done to realize the objectives of the virtual manufacturing system.

Thus, the vision is clear about VM system ; and its benefits can be attained when the system main objectives are identified. Building on the limitation of studies relevant to the VM system and lack of Arabic and Iraqi references require conducting studies in the following domains in order to clarify the conception of this new trend :

1. Identifying the job mechanism in this system by carrying profound studies in determining priorities for executing commands in each cell by using techniques of Neural Network or Fuzzy logic and the like.
2. The role of information and communication between the agents and other factories in order to realize the objectives of the virtual manufacturing.

## 2. Concept of a Proposed VM System

The vision of virtual manufacturing is to provide a capability to manufacturing in the computer. *Also a new definition of VM can be stated as a system, in which models of manufacturing objects, processes, activities, and principles evolve in a computer based environment to enhance one or more attributes of the manufacturing process [11].* A Virtual Manufacturing Enterprise is a temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks [1].

The virtual system approach takes the best of both the traditional software approach and the computing appliance approach and combines them in a way that delivers both convenience and flexibility. So Several key benefits of VS in manufacturing domain can be cited as follows: [19]

1. Service Driven Operations management: a unique software-based IT service delivery approach to enable synchronizing business and IT.
2. Sophisticated Windows, UNIX and Linux event and performance management from Windows for managing the heterogeneous enterprise
3. Centralized point of control for the network, servers, operating systems, applications and services for correlating and managing all the IT infrastructure components of a business service
4. Extensive out-of-the-box policy-based management intelligence for enhanced time-to-value
5. Superior application management architecture providing a maximum of scalability.
6. Capability to manage both Microsoft, NET and J2EE applications from the same platform.
7. Flexible management concept which allows you to configure sophisticated manager of manager concepts.

The planning and control of the flow of work through a manufacturing system is a complex task. Customer orders must be translated into orders for the many components, subassemblies and assemblies which are required to complete that order. Some components are manufactured in-house while others have to be purchased from external suppliers.

The proposed VM system represented by DB schema has been mainly designed by depending on 12 object types, 18 tables, 5 views, 1 trigger and 3 packagesThe package which includes a group of procedures or functions has been used to facilitate granting it an execution privilege by the users or agents to perform the tasks assigned to it. Figure (1) illustrates the work mechanism of the designed system according to DB schema. with regard to the roles and privileges of the proposed system, the system has been designed on the basis of 5 agents: design, process planning, MC&CE, priority planning and supervisor. Every agent (user) has its own ***roles and privileges***. The supervisor has the role of Database Administrator (DBA) which entitles him all the roles and privileges available at the Oracle because he is the General Manager. This also entitles him to give every agent roles and privileges according to the nature of their jobs. Accordingly, all the agents can work cooperatively to perform the goal of the company efficiently and securely. ***The research focuses on Priority Planning Agent procedures in Jobs sequences.***

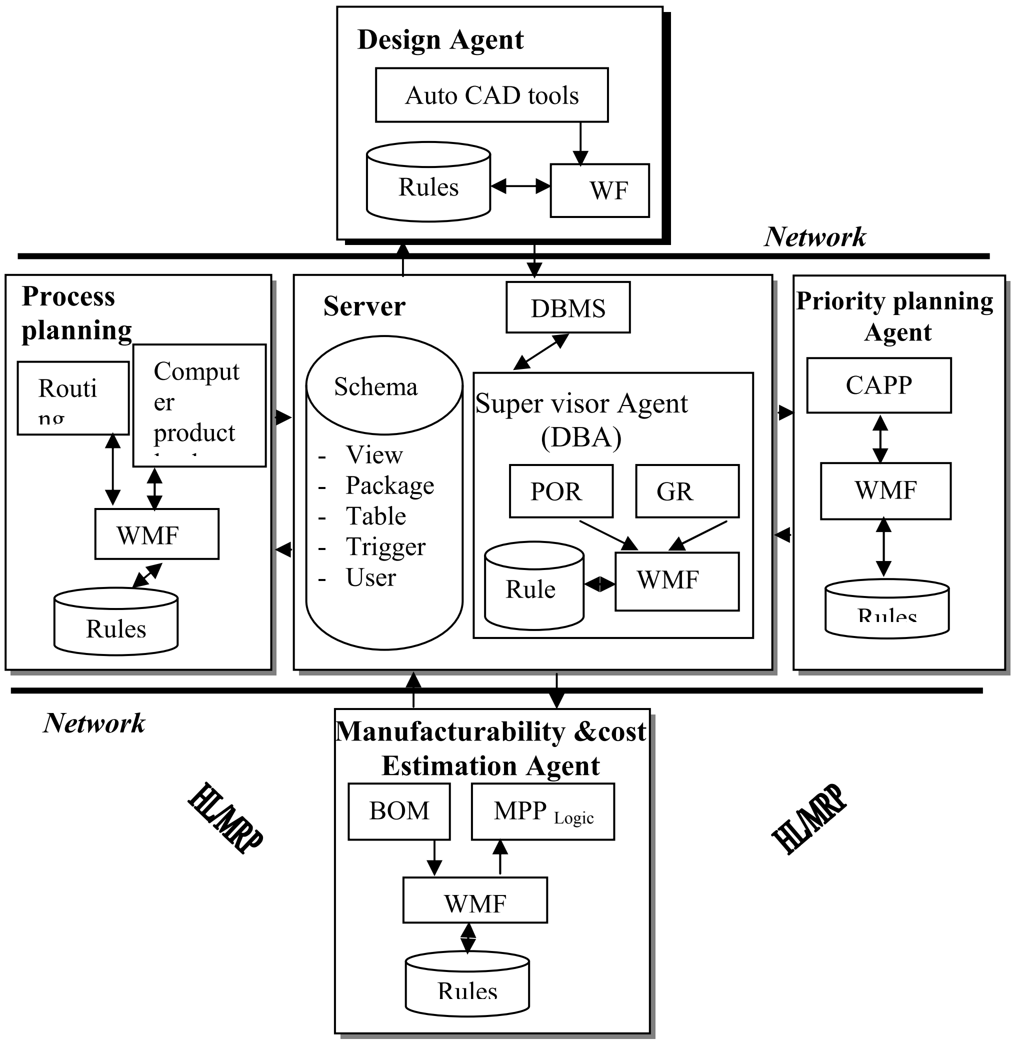


Figure (1): Mechanism of Proposed Information System.

## 2.1 Priority Planning Agent

Scheduling is defined as the problem of allocation of machines over time to complete jobs. The **m*n** job shop scheduling problem denotes a problem where a set of **n** jobs has to be processed on a set of **m** machines. Each job consists of a chain of operations, each of which requires a specified processing time on a specific machine [15]. The applied studies in this field have shown that the "*Genetic Algorithm*" is considered to be the most widely used technology in assigning machines jobs and obtaining best planning for jobs sequences on machines. The attached figure (2) is the structure of the priority-planning agent This research has adopted this style to achieve:

1. Reduction in lead-time for the manufactured products.
2. Best use (exploitation of the available resources).

## 2.2. Genetic Algorithm Representation

GA is defined as: a *computer simulation* in which generations with selections and multiplication depending on the fitness value of the evaluation of a population on computer. Better genetics of a previous generations tends to be passed on its offspring. Here, selection means the survival of the fittest, while multiplications are the processes in that organisms' multiply. GA works best in the following situations [9]:

1. Potential solutions can be represented in a way which exposes components of solutions.
2. Operators to mutate and hybridize these representations are available

GA has been designed as general-purpose optimization methods [21]. It is a search algorithm that uses operations found in natural genetics to guide the trek through a search space . In GA, each solution is represented in the form of a finite length array called chromosome. [7].

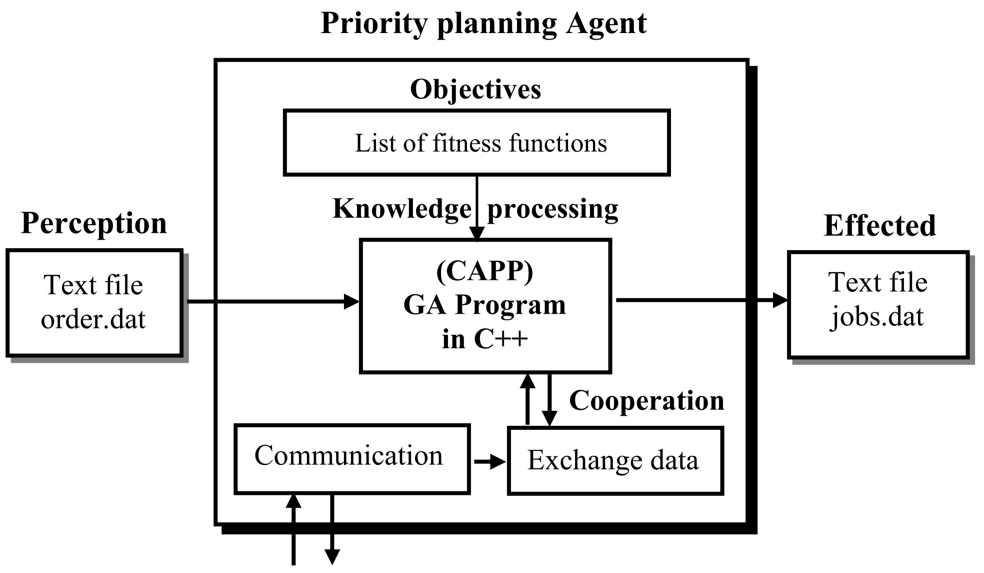


Figure (2): The Structure of the Priority Planning Agent

A chromosome is composed of a set of locations known as genes that assume discrete values pertaining to the problem's solution. A critical point when applying GA to an optimization problem is to find a suitable coding scheme that transforms feasible solutions into representations amenable to a GA search and reversibly decodes those representations [22].

## 2.3 Job Sequencing

Jobs sequence, time ordered sequence of the important actions is taken by one or more transitions. Jobs sequence where the operations of each transaction is executed consecutively without any interleaved operations from other transactions. No guarantee that results of all serial executions of a given set of transactions will be identical. Manufacturing jobs sequencing is an optimization process. It has a significant impact on the efficiency and productivity of any manufacturing enterprise. Job sequencing problems are typically NP-hard; that is, it is impossible to find an optimal solution without the use of an essentially enumerative algorithm and the computation time increases exponentially with problem size. Manufacturing jobs sequencing is one of the most difficult manufacturing problems. The jobs sequencing problem is further complicated when unforeseen dynamic situations arise on the shop floor. Because of its dynamic nature and its practical interest for industrial applications, the manufacturing jobs sequence problem has been the subject of extensive technology. However there are a number of features and benefits to decide the sequence of jobs [13] [16]:

1. The resulting software will take manufacturing process plans as inputs and automatically generate a manufacturing jobs sequence for a shop floor, a plant or an enterprise, with estimation of production cost and delivery options. This will allow the manufacturing enterprise to respond to customer requests quickly and therefore to win in the increasingly competitive market.
2. Jobs sequencing is done through negotiation among intelligent agents representing related manufacturing resources, rather than centralized. Such a jobs sequencing system can quickly respond to changes in the shop floor through dynamic jobs sequencing.
3. Developing an agent-based intelligent job sequencing system using advanced technologies including Intelligent Agents, Genetic Algorithm, Internet, and Web technologies. This will allow managers to monitor the production status anywhere and to jobs sequencing a customer order over the Web.
4. Remoting an online access to the manufacture database. Allowing for prompt customers awareness about the anticipated due dates of their orders assumes developing efficient runtime and quality of solution. Job sequence routine for planning the assignment of the submitted customers orders to the factory's machines.

## 2.4 Genetic Algorithm Design for Job Sequencing

The success of GA strongly depends on how to map the concrete representations toward the abstract searching space in which operators move through [22]. In fact the jobs sequencing is an NP hard problem and that for practical problems in higher dimensions the best solution can only be found by trial and error, so it can use a modeling concept which aims at creating an environment which allows the application of optimization techniques like Genetic Algorithms in dynamic and complex environments. A Genetic Algorithm is an iterative procedure which usually maintains a constant population size. Termination is commonly triggered by reaching a maximum number of generations, or by finding an acceptable solution [4]. Probability of mutation, probability of crossover, and population size, either empirical or trial and error method are used according to the features of problems. [20]. During iteration t, the GA maintains a population P(t) of solutions $x^t_1$. .$x^t_R$ (the population size R remains fixed).Each solution, $x^t_i$ is evaluated by a function with $E(x^t_i)$ being a measure of the fitness of the solution. The fitness value determines the relative ability of an individual to survive and produce offspring in the next generation. In the (t+1) th iteration a new population is formed on the basis of the genetic operators selection, crossover and mutation [7].

## 2.5 Problem Statements

Customer order has N products (jobs) $J_1$, $J_2$, …, $J_N$. So each chromosome contains N genes (alleles), each gene represents job $J_i$, in the job sequencing problem, genes of each chromosome are set to a random numbers within the range 0..N-1, with given processing time $P_1$, $P_2$, …, $P_N$, and due dates $DD_1$,$DD_2$, …, $DD_N$. These data are stored in text file by oracle, double click release order button from order form, the *job_schd.exe* program written in C++ will be executed to read this file for CAPP purpose, jobs sequenced depending on: minimum Average Completion Time (ACT), maximum Utilization (U), maximum Average number of Jobs in the System

(AJS), or minimum Average Job Lateness (AJL) [6]. So the chromosome has N genes, each gene represents a specific job in the customer order. Each job $J_i$ (for i=1 to N), consists of a chain of operation on a specific machines determined by a process plan called path technology or routing. Each machine m has setup time and operation time. Machine process time represents the summation of them.

$$PT_m = Setup\_Time_m + Operation\_Time_m \qquad For\ m = 1\ to\ M\ \ldots (1)$$

And process time for job i ($J_i$), PT(i) is computed as follows:

$$PT(i) = \sum_{m=1}^{R} PT_m \qquad Where\ R\ is\ chain\ of\ machines\ J_i\ routing\ \ldots(2)$$

The customer may need one or more units of job $J_i$ in his order, call it D(i) demand for product $J_i$, the final processing time $P_i$ for a specific job $J_i$ computed as follows:

$$P(i) = PT(i) \times D(i) \qquad For\ i = 1\ to\ N \qquad \ldots (3)$$

Where:

| | |
|---|---|
| *D: Job demand* | *J&K: Two crossing sites* |
| *DD: Job due date* | *M: Number of machines in the factory* |
| *FT : Job flow time* | *N: Number of jobs in the customer order* |
| *JL : Job lateness* | *PT: Process time for one product* |
| *P: Job process time* | *R: Product path technology (routing)* |
| *$P_{size}$: Population size* | *Child: Offspring* |

## 4. A Proposed procedure of GA

1. The population size is determined as follows:

$$P_{size} = \begin{cases} 2 & \text{If } N < 5 \\ 6 & \text{Otherwise} \end{cases} \qquad \ldots (4)$$

2. It is difficult to choose suitable parameters for a GA and it is still important to enhance the performance (solution quality and search efficiency) of GAs [23]. Convergence of the algorithm is dependent on a suitable choice or careful selection of these parameters which are related to the complexity of the problem [12]. These parameters are as follows:

   Probability of mutation $P_m$ is 2/N.

   Probability of selection $P_s$ is 2/pop_size.

3. The termination criteria (stopping parameter) are run time, and fitness of the best individuals. GA is used for job sequencing problem basically works in 20 iterations, after finishing, the program asks the user if he wants to continue searching (another 20 iteration) for another best result.

   The proposed GA procedure for job sequence is described as follows:

   Integrated example about Individual chromosome :

| *Job No. (Gene)* | *Job Sequence* | *Process Time (Days)* | *Flow Time (Days)* | *Due Date (Days)* | *Job Latenes. (Days)* |
|---|---|---|---|---|---|
| 0 | **Ashor_Ellipse_Melamen_Desk_240*120_Cm** | 1 | 1 | 2 | 0 |
| 1 | **Warkaa_Melamen_Desk_Appendix_120*40_Cm** | 1 | 2 | 1 | 1 |
| 2 | **Warkaa_Melamen_Desk_150*80_Cm** | 2 | 4 | 3 | 1 |
| 3 | **3_Drawer_Teak_Desk_150*80_Cm** | 4 | 8 | 2 | 6 |
| 4 | **Ashor_Teak_Desk_Appendix_120*40Cm** | 5 | 13 | 2 | 11 |
| 5 | **Ashor_Ellipse_Teak_Desk_240*120_Cm** | 5 | 18 | 2 | 16 |
| 6 | **4_Drawer_Teak_Desk_150*80_Cm** | 7 | 25 | 3 | 22 |
|  | **Total** | 25 | 71 |  | 57 |

**Chromosome 1:**

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

**The Four Fitness Functions are:**

- **Average Completion Time (ACT) : 71 / 7 = 10.143 Days.**
- **Utilization (U) : 25 / 71 = 35.211 %.**
- **Average No. of Jobs in the System (AJS) : 71 / 25 = 2.840 Days.**
- **Average Job Lateness (AJL) : 57 / 7 = 8.143 Days.**

## 4.1 Initialization

The GA initialization process, in which a set of initial random solutions are generated, is conducted using a randomized version. So the initial population (chromosomes) of individuals (solution candidates) is generated randomly or heuristically. During each iteration step, called a generation, the individuals in the current population are evaluated and given a fitness value. In order to form a new population, individuals are selected with a probability proportional to their relative fitness which ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population [3].

a. First generated chromosome has the same jobs sequence in the order, second chromosome has reverse order (crom1=a, b, c and crom2=c, b, a).
b. When N > 2, the next step is to generate the rest chromosomes randomly according to the following algorithm:

```
{
   Randomize ();
   For i = 2 to i < $P_{size}$ do          /* Psize = Population size */
/* i is the index of chromosome */
    {
     Label1:For j = 0 to j < N do   /* N is chromosome length */
       {
          Label2:    k := rand() % N;  /* K is the gene */
          For g = 0 to g < j do
     If k = crom[i].gen[g] goto Label2; /* check duplicated gene */

                                     crom[i].gen[j] := k;
       }
     For c = 0 to c < i do  /* check for unrepeated chromosome */
            If crom[i] = crom[c] goto Lable1;
      }          }
```

## 4.2 Evaluation of Fitness

The fitness of each chromosome is evaluated by substituting the values of decoded variables into fitness function [4]. An evaluation or fitness function plays the role of the environment in distinguishing between good and bad solutions [7]. Sequencing jobs in work centers according to one of effectiveness measures is used as a fitness function. The decision-maker or CAPP user selects one of the following four constructed fitness functions according to customer opinion.

1- Minimized Average Completion Time (ACT).

$$ACT = \left(\sum_{i=1}^{N}\sum_{j=1}^{i} P(j)\right) \Big/ N \qquad \text{.... (5)}$$

2- Maximized Utilization (U).

$$U = \left(\sum_{i=1}^{N} P(i) \Big/ \sum_{i=1}^{N}\sum_{j=1}^{i} P(j)\right) \times 100 \qquad \text{.... (6)}$$

3- Maximized Average number of Jobs in the System (AJS).

$$AJS = \left(\sum_{i=1}^{N}\sum_{j=1}^{i} P(j)\right) \Big/ \sum_{i=1}^{N} P(i) \qquad \text{.... (7)}$$

4- Minimized Average Job Lateness (AJL).

$$AJL = \left(\sum_{i=1}^{N} JL(i)\right) \Big/ N \qquad \text{.... (8)}$$

Job Lateness JL for job i $J_i$ calculated as follows:

$$JL(i) = \begin{cases} FT(i) - DD(i) & \text{If } FT(i) > DD(i) \\ 0 & \text{Otherwise} \end{cases} \qquad \text{.... (9)}$$

Flow Time FT is the amount of time a job spends in the simulated production system [Patrick, 2000, p.2661]. Flow time for $J_i$, FT(i) calculated as follows:

$$FT(i) = \sum_{j=1}^{i} P(j) \qquad \textit{For } i = 1 \textit{ to } N. \qquad \text{.... (10)}$$

## 4. 3 Selection

Selection is the process of searching for parents. Selection procedure alone cannot produce any new solution candidates in the search space [18], but it gives a straightforward way of choosing offspring for the next generation. Select return the population index value corresponding to the selected individual [2] [5]. We suggest a binary selection which is a robust, commonly used mechanism that provides a good convergence rates, avoiding a sub optimal solutions, and proved to be efficient and simple to code. The algorithm is as follows:

```
{ /*  select two chromosome (K and L) randomly  */
  k := rand() % Psize;
  Do { l := rand() % psize;   } while ( k == l);
  Switch (fitness)
     {
 Case 'U' , 'AJS' : /*  U: utilization , AJS : Avg. No. of  jobs in the system  */
                     If ( crom[k].fitness > crom[l].fitness) then
                            Return (k);
                     Else
                            Return (l);
                     Break;
 Case 'ACT' , 'AJL' : /*  Act : Avg. compilation time , AjL: Avg. job lateness  */
                     If ( crom[k].fitness < crom[l].fitness) then
                            Return (k);
                     Else
                            Return (l);
                     Break;
      }          }
```

## 4.4 PMX Crossover Operator Mechanisms and Design

Crossover is the most important reproduction operator to find new starting points for search. It takes two individuals, called parents selected by call select function two times, and produces one or two new individuals (in our case produce one) called the offspring by swapping parts of the parents. Crossover combines two chromosomes (parents) to produce a new chromosome (offspring). That new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. In its simplest form the operator works by exchanging substrings after a randomly selected crossover point [21]. The crossover operator, which combines the features of two parent structures to form two similar offsprings; this is applied under a random position cross with a probability of performance (the crossover probability) Pc [7]. Crossover operators are an essential part of GA as they help in inheriting better characteristics from the fittest solutions among generations [22].

There are two schemes of Crossover. One is Uniform Crossover in which a child genome is created with its own genetics that consists of genetics passed from one of parents with some probability p and those from another one with the probability 1−p. Each genotype is occupied by the genes from parents independently. Another scheme is 2-point Crossover that parents genomes whose forms are not strings but rings are split into 2 parts at two cross points chosen at random, and then combined to make a new genome. The different point between them is that 2-point Crossover might suppress some increase of its action rather than Uniform Crossover. Diversity is to have high probability of a production of an interesting child genome is, however, assured by Uniform Crossover rather than 2-point Crossover. The problem which remains and to be expected is how to update the population. Generally, C children produced from C parents replace all the parents. Then, the next generation becomes totally new. This scheme is called spawning. The diversity of the population depends, however, strongly on the selection of the parents [24].

Partially Mapped Crossover PMX is a kind of two-point crossover operation. Delivery jobs sequence for a selected set of jobs will be exchanged between two parent solutions. In this case, the delivery jobs sequence or production decision for each job is defined in the parent solutions. In GA implementation, the horizontal exchange of rows between two different solutions as the crossover operator is used. This is done by randomly selecting any number of job sequences and exchanging their delivery amounts between the two solutions to generate two new child solutions [22]. Goldberg and Lingle in 1985 have suggested PMX operator. Under PMX, two strings (permutations and their associated alleles) are aligned, and two crossing sites (j and k) are picked uniformly at random along the strings. The two crossing points define a matching section that is used to effect cross through position-by-position exchange operations. Consider two strings [5]:

| | | |
|---|---|---|
| P1 = 9 8 4 | 5 6 7 | 1 3 2 10 |
| P2 = 8 7 1 | 2 3 10 | 9 5 4 6 |
| | J | K |

PMX proceeds by position wise exchanges. First, map string p2 to string p1, the 5 and 2, the 3 and the 6, and the 10 and the 7 exchange places. Similarly map string p1 to string p2. Following PMX are left with two offspring 1 and 2 where each string contains information partially determined by each of its parents: In which duplicate batch numbers appeared in bold.

| | | |
|---|---|---|
| Offspring1 = 9 8 4 | 2 3 10 | 1 **3** **2** **10** |
| Offspring2 = 8 **7** 1 | 5 6 7 | 9 **5** 4 **6** |
| | J | K |

The two new strings then follow another treatment: if a duplicate batch number is found in string 1, it is swapped with the batch number of string 2 in front of the first duplicate batch position of string 1 (for instance 3 in string 1 is exchanged with 6 in string 1). The same procedure applies for duplicate batch numbers of string 2. The process is continued until all duplicates have been removed. The offsprings finally obtained are:

| | | |
|---|---|---|
| Offspring1 = 9 8 4 | 2 3 10 | 1 6 5 7 |
| Offspring2 = 8 10 1 | 5 6 7 | 9 2 4 3 |
| | j | k |

The PMX crossover algorithm shows that p1 and p2 are two parents, child is the new generated chromosome (offspring), and t variable represents the gene position (index) in parent p1 where its value is equal to the selected gene from parent p2.

```
{
    j := rand() % N;  /* N is chromosome length */
    Do {k := rand() % N;} while (j == k);
    /* select 2 crossing points (j and k) randomly */
          If (j > k) then
           {   i := j;     j := k;     k := i;        }
               Strcpy(child.gen , crom[p1].gen);
               For i = j to i <= k do
              {
               c := child.gen[i]; /* same as child.gen[i] = crom[p2 ] gen[i] */
               t := strchr(child.gen , crom[p2].gen[i]) – child.gen;
                              child.gen[i]:=child.gen[t];
               /*      same      as      child.gen[i]      :=crom[p2].gen[i];
               child.gen[t] :=
              }              }
```

### 4.5 Binary Mutation Operation

Mutation is the process of modifying genes of the chromosome itself. It arbitrarily alters one or more components of a selected structure so as to increase the structural variability and robustness of the population. Each position of each solution vector in the population undergoes a random change according to a probability defined by the mutation rate (the mutation probability) $P_m$ [7]. Mutation is essentially an arbitrary modification introduced to prevent premature convergence to local optima by randomly sampling new points in the search space. In the case of bit strings, mutation is applied by simply flipping bits randomly in a string with a certain probability called mutation rate [4].

Mutation operators are applied to each child solution resulting from the crossover operation. They help the GA to reach further solutions in the search space. The idea of the mutation operation is to randomly mutate a solution's genes (the values assigned to each cell in our two-dimensional matrix structure) and hence produces a new and better solution [22].

Proposed for mutation a simple swapping of two unique elements randomly selected in the offspring. The mutation operator is designed to conduct delivery exchanges in a random fashion. Mutation procedure uses the function flip to determine whether or not to exchange two genes selected randomly from the new chromosome (child) that is generated by crossover recently [5]. Child is the input to the procedure, the exchange occurs two times in it, and the algorithm is as follows:

```
{
  For m =1 to m = 2 do /* two mutations for each child */
  Do { /*  select two points ( j and k) randomly */
   j := rand() % N; /* N is chromosome length */
     Do {   k := rand() % N;     }  while ( j == k);
     Do {   pm := rand() % 10; }  while ( pm == 0); /* 0 < pm < 10 */
            pm := pm / 10;       /* 0 < pm < 1 */
            If flip(pm) then
              {
                a := child.gen[j];
                child.gen[j] := child.gen[k];
                child.gen[k] := a;
                break; /* Exit from the first loop do …. While (true) */
              }
          } while (TRUE);
}
```

The input to flip function is pm variable, the algorithm is as follows:

```
{
   Do {   ff := rand() % 10;   }  while ( ff == 0);
   ff := ff / 10;
   If ( ff <= pm ) then  Return (TRUE);   Else   Return (FALSE);
}
```

### 4.6 Replacement

Replacement is the use of a new generated population for a further run of algorithm.

```
{
  For i := 1 to $P_{size}$      do
      crom[i].gen := child[i].gen;
}
```

### 4.7 Test

After 20 iterations, the program displayed the best overall optimum solution found to that point, and asked the user if he wanted to continue searching, try to find a better optimum solution, starting from latest new population. Else, at the completion of the program, the final result saved in text file *jobs.dat* to store it in oracle database later. In case of performing sequence of jobs for the first time, they will be stored directly in the *job_sched* database table. If the optimal solution obtained is better than the stored one, it will be updated otherwise, the database table will stay unchanged.

## 5. Priority Planning Agent Interface

The P.P. agent prepares the *order.dat* file which includes the customer order and the process time as well as the due date for each product. The file is then sent (as a message) through the interface (which is the DDE) to enter C++ software and execute the program *job_schd.exe* specialized in GA which in its turn reads *order.dat* file then it selects the fitness function to decide the job sequences by the P.P. agent. The optimal solution lies in storing the *jobs.dat* file then exiting from GA execution and going back to oracle form so that it will be possible to read the file and update the database according to that result..

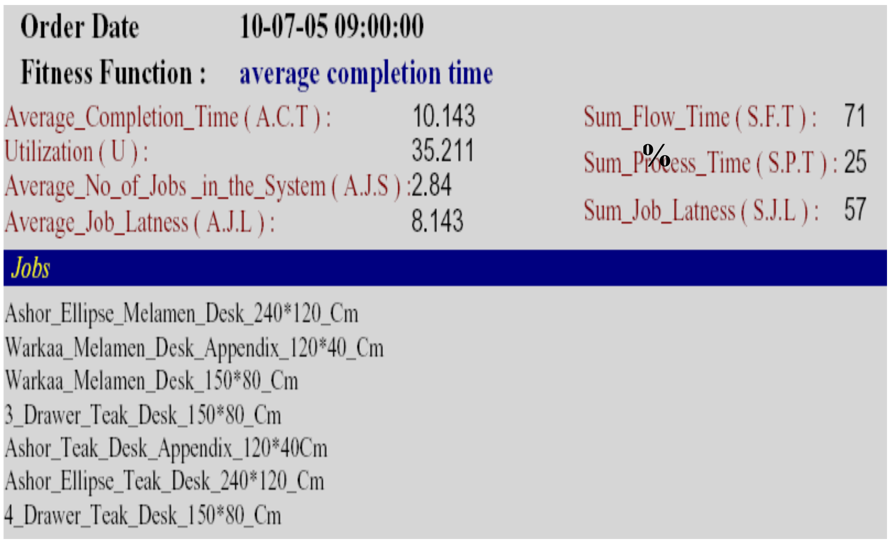
## 6. Discussion and Analysis of the Application Results

The Priority Planning Agent receives the *order.dat* file from the process Planning Agent. This file contains the required information about the orders and is considered as one of the main inputs for those software designed by the use of C++ language according to the genetic algorithm logic for planning job priorities in the shop floor.

By cooperation with the *Supervisor Agent,* one of the rules relied on in designing software to plan for job priorities is selected. The rule of the ***Average Completion time*** has been selected to be a fitness function according to which the optimal sequence for executing the manufacturing planned jobs has also been selected. This rule aims at reducing the lead time for products' manufacturing and achieving quick response for the customers' orders.

1. According to what has been mentioned above, table (1) illustrates the softwares execution results according to ***ACT*** function. The execution outlets a certain sequence for the jobs each time. According to this sequence the value of the fitness function is estimated. The best sequence for jobs executions has been selected according to the lowest value of fitness function of 10.143 as explained in the Table (1).

Table (1) The Optimal Job Sequence According to Average Completion Time (ACT).



**Order Date** 10-07-05 09:00:00

**Fitness Function :** average completion time

| | | | |
|---|---|---|---|
| Average_Completion_Time ( A.C.T ) : | 10.143 | Sum_Flow_Time ( S.F.T ) : | 71 |
| Utilization ( U ) : | 35.211 | Sum_Process_Time ( S.P.T ) : | 25 |
| Average_No_of_Jobs _in_the_System ( A.J.S ) : | 2.84 | Sum_Job_Latness ( S.J.L ) : | 57 |
| Average_Job_Latness ( A.J.L ) : | 8.143 | | |

*Jobs*

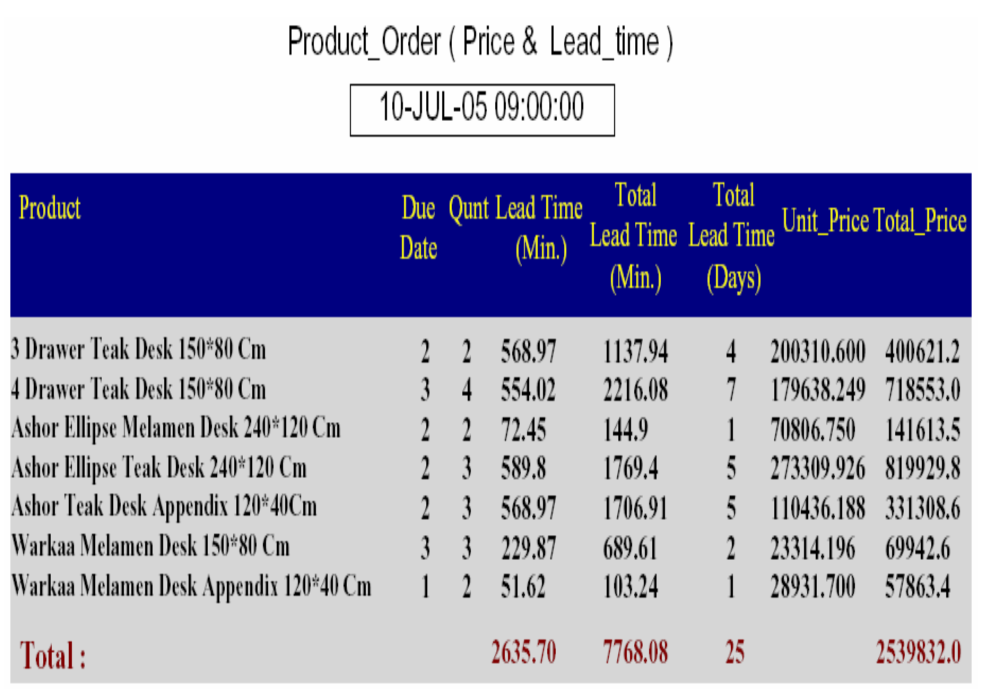
Ashor_Ellipse_Melamen_Desk_240*120_Cm
Warkaa_Melamen_Desk_Appendix_120*40_Cm
Warkaa_Melamen_Desk_150*80_Cm
3_Drawer_Teak_Desk_150*80_Cm
Ashor_Teak_Desk_Appendix_120*40Cm
Ashor_Ellipse_Teak_Desk_240*120_Cm
4_Drawer_Teak_Desk_150*80_Cm

The execution results are naturally stored in the database specialized in the order; namely in the *Job_Shed* Table. It is worth to mention here that in case an execution repetition for the same order and for the same objective function takes place, the planned software will have the capacity of changing the stabilized result in the file with a new fitness function if this new fitness function has a lower value, otherwise the stabilized value remains unchanged in the file.

3. After calculating the order's lead time with the products involved, estimating the product's costs and consequently the order's total cost, the customer is informed with all this information. As soon as the customer agrees on this, the Supervisor Agent will issue an order release for the product to be manufactured in the factory. The table (2) explains the manufacturing order's Price and Lead time.

It has been shown so far that the basic idea of our system design and its application is to manufacture a product by the use of the computer, as procedures end as soon as commands are given to manufacture an order at the shop floor inside the factory.

Table (2): Manufacturing Order's Price and Lead-time.



Product_Order ( Price & Lead_time )

10-JUL-05 09:00:00

| Product | Due Date | Qunt | Lead Time (Min.) | Total Lead Time (Min.) | Total Lead Time (Days) | Unit_Price | Total_Price |
|---|---|---|---|---|---|---|---|
| 3 Drawer Teak Desk 150*80 Cm | 2 | 2 | 568.97 | 1137.94 | 4 | 200310.600 | 400621.2 |
| 4 Drawer Teak Desk 150*80 Cm | 3 | 4 | 554.02 | 2216.08 | 7 | 179638.249 | 718553.0 |
| Ashor Ellipse Melamen Desk 240*120 Cm | 2 | 2 | 72.45 | 144.9 | 1 | 70806.750 | 141613.5 |
| Ashor Ellipse Teak Desk 240*120 Cm | 2 | 3 | 589.8 | 1769.4 | 5 | 273309.926 | 819929.8 |
| Ashor Teak Desk Appendix 120*40Cm | 2 | 3 | 568.97 | 1706.91 | 5 | 110436.188 | 331308.6 |
| Warkaa Melamen Desk 150*80 Cm | 3 | 3 | 229.87 | 689.61 | 2 | 23314.196 | 69942.6 |
| Warkaa Melamen Desk Appendix 120*40 Cm | 1 | 2 | 51.62 | 103.24 | 1 | 28931.700 | 57863.4 |
| Total : | | | 2635.70 | 7768.08 | 25 | | 2539832.0 |

Accordingly, the application process represents assimilation to the actual manufacturing environment under study. As it has been explained before two orders which have, to some extent, some different features have been chosen. In order to explain the capabilities of the designed system at different stages of manufacturing, and within the framework of integration between (**CAD/CAPP/HLMRP**) and after all for the sake of implementing the philosophy of the virtual manufacturing, each order consists of a number of products. In the same way and concept the network, designed by the use of window server 2003 to the server and the window XP assigned to the clients, has a great role in achieving the concept of the VM. This has been demonstrated by the efficient flow of information and data throughout the manufacturing environment.

While the application has embodied the second objective of the thesis, it has, at the same time, provided integrated information to attain the other objectives. The application results will be analyzed within the frame of the following domains.

## 7. Conclusions

The results of the designed system application have revealed that the operations planning by the use of the GA philosophy will perform a great role in calculating the product's lead time at the manufacturing operations' stages. This role supports the VM philosophy in calculating the industrial part of the products lead time quickly. The following points have been observed on executing the GA software:

1. The application results have confirmed that the designed *GA software* efficiency depends upon the *number **of jobs available at the time of execution***; whenever the number of jobs is bigger, the software execution efficiency is better.
2. If we have to wait for two jobs only then the Job_Schd.exe program will choose the best fitness because there are only two probabilities without going to (entering) the GA. But if the number of jobs is equal to three or four then the population size = 2, otherwise the population size = 6.
3. The designed software includes (Job_schd.exe) according to GA logic in programming four rules from the rules special to the job priorities. The execution process of the system has confirmed to be easy to add new rules or to include several rules in one new rule and execute it on many jobs by giving very accurate results.
4. It has been confirmed by execution that the (Job_schd.exe) program has the ability to reexecute the same order and for the same fitness function in case that the reexecution results are better according to the adopted fitness function; the stored result in the database will be executed by the new value, otherwise the stored result will stay as it was.
5. The execution results of the (Job_schd.exe) program concerning the job sequencing for the first order with seven jobs confirmed two of the jobs sequence:

Ashor_Ellipse_Melamen_Desk_240*120_Cm.,
Warkaa_Melamen_Desk_Appendix_120*40_Cm.,
Warkaa_Melamen_Desk_150*80_Cm.,
3_Drawer_Teak_Desk_150*80_Cm.,
Ashor_Teak_Desk_Appendix_120*40Cm.,
Ashor_Ellipse_Teak_Desk_240*120_Cm.
4_Drawer_Teak_Desk_150*80_Cm.
And

Warkaa_Melamen_Desk_Appendix_120*40_Cm.,
Ashor_Ellipse_Melamen_Desk_240*120_Cm.,
Warkaa_Melamen_Desk_150*80_Cm.,
3_Drawer_Teak_Desk_150*80_Cm.,
Ashor_Teak_Desk_Appendix_120*40Cm.,
Ashor_Ellipse_Teak_Desk_240*120_Cm.,
4_Drawer_Teak_Desk_150*80_Cm.

have achieved optimal solution for two fitness function namely ACT (Min.) = 10.143 days and U (Max.) = 35.211%, see appendix (E). As this last one has achieved Average job lateness (AJL (Min.) = 8 days) less than the first sequence it has been chosen in executing the order.

## REFERENCES

[1] Camarinha-Mates, L.M. and others, ***Towards an architecture for virtual enterprises***, Journal of Intelligent Manufacturing, vol. 9, No. 2, 1997.

[2] Chris, M. and Jimmie, Browne, ***CADCAM principles, practice and manufacturing management***, 2nd edition, Addison Wesley, 1998.

[3] George, Koch and Kevin, Loney, *Oracle 8:* ***The complete reference***, Osborne, MC Graw-Hill, 1997.

[4] George, W. ; Michael A.; Alois R. and Stefan W., ***MODELLING OF AN AGENT-BASED SCHEDULE OPTIMISATION SYSTEM***, 2004, @Profactor.at, @cast.uni-linz.ac.at.

[5] Goldberg, D. E., ***Genetic Algorithms in Search, Optimization****, and Machine, Learning*, Addison-Wesley, 1989.

[6] Heizer, j., and Render, B., ***Principles of operations management***, Prentice Hall, Inc., 1999.

[7] Herrera, F.; M. Lozano, and J. L. Verdegay, ***Tuning Fuzzy Logic Controllers by Genetic Algorithms***, International Journal of Approximate Reasoning, 12:299-315, 1995.

[8] Highsmith, J., ***What is agile software development?*** The Journal of Defense Software Engineering, October 2002.

[9] Hondroudakis, A.; Malard,J., and Wilson, *G.V.A* ***An Introduction to Genetic Algorithm Using RPL2 EPCC TRACS***, Student note, 1995.

[10] Junior, L.O. and others. ***An approach to design control systems for distributed production system as collaborative architecture*** [icaps03.itc.it/satellite_events/documents/dc/19/Oliviera.pdf].

[11] Nagi, R., and Song L., ***Design and implementation of a virtual information system for agile manufacturing***, IIE Trans. on Design and Mfg. (vol. 29, No. 10), p. 839 - 859, 1997.

[12] Nasir, H., and Andrew W., ***Genetic Algorithms and Machine Scheduling With Class Setups***, International Journal of computer and engineering management, 1997. At the site: http://www.itrssm@au.ac.th.

[13] Patrick, R.; Peter, T., and Gregory, V., ***Using genetic algorithms to solve the multi-product JIT sequencing problem with set-ups***, International journal of production research, Vol. 38, No. 12, p 2653-2670, 2000.

[14] Paulo, L. and Restivo, F., ***An agile and cooperative architecture for distributed manufacturing systems***, 2001, [citeseer.ist.psu.edu/459153.html].

[15] Roberto, C., and Weiming, Sh., ***Agent-Based Distributed Manufacturing Scheduling***, January 2002, [http://www.nrc.ca/imti/].

[16] Pinedo, M., ***Scheduling Theory: Algorithms and Systems***, Prentice Hall, Second Edition, 2002.

[17] Shahrokh, M. and Chu, R. ***Agile manufacturing an enabling technology for competing in changing markets.*** www.semiconductorfabtech.com/journals/edition.98.

[18] Sung-Chung Kim and Kyung-Hyun Choi, ***Development of Flexible Manufacturing System using Virtual Manufacturing Paradigm***, International Journal of the Korean Society of Precision Engineering, Vol. 1, No. 1, June 2000.

[19] Slater, M., and Others , Acting in Virtual Reality: [http://www.cs.ucl.ac.uk/staff/m.slater/vr/Projects/Acting]

[20] Tae Hyong Chong, and Joung Sang Lee, ***A Design Method of Gear Trains Using a Genetic Algorithm***, International Journal of the Korean Society of Precision Engineering, Vol. 1, No. 1, June 2000.

[21] Takeshi, Y., and Ryohei N., ***A genetic algorithm with multi-step crossover for job-shop scheduling problems***, International conference on genetic algorithms in engineering systems: innovations and applications, Pp. 146-151, 1995.

[22] Tamer, F. Abdelmaguid, Maged, M. Dessouky, ***A genetic algorithm approach to the integrated inventory distribution problem***, 2004, http://www-rcf.usc.edu/~maged/publications/GAinventoryrouting.pdf, Email: maged@usc.edu

[23] Wang, L., Zheng, D., ***A modified genetic algorithm for job shop scheduling***, International journal of advanced manufacturing technology, 20:72-76, 2002.

[24] Yamaguchi, A., ***Genetic Algorithm for SU(N) gauge theory on a lattice***, Ochanomizu University, 2-1-1 Otsuka Bunkyoku Tokyo, Japan, 1998.