CHAPTER - 3

AI SEARCH ALGORITHMS AND SOFTWARE DEVELOPMENT

3.1 Introduction

In manufacturing industry software provides all the business management tools needed, including: forecasting, sales order automation, production management, capacity management and Customer Relationship Management etc., all on a single platform. The solution makes extensive use of automation, so the essential transactions of manufacturing activities are generated and customized with relevant data as soon as the set stage in process is reached. The manufacturing industry has seen dramatic changes (customers have demands and they are dynamic in product innovation) forcing the compression of product life cycles. The software embedded in these manufacturing activities and systems has driven this innovation, but is challenged by development complexity.

Consequently, picking the right integrated manufacturing software is crucial. In order to ensure the survival and growth of any manufacturing industry it is imperative to identify the manufacturing software programs providing the lowest possible total cost of ownership with quick return on the investment. Hence, the selection of software package is likely to be one of the most critical capital expenditure decisions a manufacturing company should make.

3.2 AI Search Algorithms

Artificial intelligence (AI) is the intelligence of machines and the branch of computer science that aims to create it. AI textbooks define the field as "the study and design of intelligent agents" where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success.

AI research is highly technical and specialized, deeply divided into subfields that often fail to communicate with each other. Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers. AI research is also divided by several technical issues. There are subfields which are focused on the solution of specific problems, on one of several possible approaches, on
the use of widely differing tools and towards the accomplishment of particular applications. The central problems of AI include such traits as reasoning, knowledge, planning, learning, communication, perception and the ability to move and manipulate objects. General intelligence (or "strong AI") is still among the field's long term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are an enormous number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. Artificial intelligence has been the subject of optimism, but has also suffered setbacks and, today, has become an essential part of the technology industry, providing the heavy lifting for many of the most difficult problems in computer science.

3.2.1 Best First Search

Best first search is an instance of the general Tree search or Graph search algorithm in which a node is selected for expansion based on evaluation function. Traditionally the node with the lowest evaluation is selected for the expansion because the evaluation measures distance to the goal. Best first search can be implemented within general search framework via a priority queue, a data structure that will maintain the fringe in ascending order of values. This search algorithm serves as combination of depth first and breadth first search algorithm. Best first search algorithm is often referred to as greedy algorithms because they quickly attack the most desirable path as soon as its heuristic weight becomes the most desirable.

3.2.2 A* Algorithm

A* algorithm is a general search algorithm that is extremely competitive with other search algorithms and yet intuitively easy to understand and implement. It generates and process the successor states in a certain way whenever it is looking for the next state to process. A* algorithm employs a heuristic function to try to pick the best State to process next. If the heuristic function is good, not only will A* algorithm find a solution quickly, but it can also find the best solution possible.

The time complexity of A* algorithm depends on the heuristic. In the worst case, the number of nodes expanded is exponential in the length of the solution (the shortest path), but it is polynomial when the search space is a tree, there is a single goal state.
The main drawback of A* algorithm, and indeed of any best-first search is its memory requirement. Since at least the entire Open list must be saved, A* algorithm is severely space-limited in practice, and is no more practical than breadth-first search on current machines. For example, while it can be run successfully in the Eight Puzzle, it exhausts available memory in a matter of minutes on the Fifteen Puzzle.

### 3.2.3 Hill Climbing Search

Hill climbing search algorithm is simply a loop that continuously moves in the direction of increasing value, which is uphill. It stops when it reaches a “peak” where no neighbourhood has a higher value. The hill climbing comes from an idea that if you are trying to find the top of the hill and you go up the direction from wherever you are. The question that remains is whether this hill is indeed the highest hill possible. Unfortunately, without further extensive explorations, that question cannot be answered. This technique works but as it uses local information, that’s why it can be misleading. This algorithm does not maintain a search tree, so the current node data structure need only record the state and its objective function value. In this algorithm only a local state is considered when making a decision of which node to expand next. When a node is entered, all of its successor nodes have a heuristic function applied to them. The successor node with the most desirable result is chosen for traversal. Hill climbing sometimes called greedy local search because it catches a good neighbourhood state without thinking ahead about where to go next. Hill climbing often makes very rapid progress towards a solution because it is usually quite easy to improve a bad state. Hill climbing is best suited to the problems where the heuristic gradually improve the closer it gets to the solution it works badly where there are sharp drop-offs. It assumes that local improvement will lead to global improvement.

### 3.2.4 Means-Ends Analysis (MEA)

Means-Ends Analysis (MEA) is a technique used in Artificial Intelligence for controlling search in problem solving computer programs. It is also a technique used since 1950s as a creativity tool, most frequently mentioned in engineering books on design methods. Means-Ends Analysis is also a way to clarify one's thoughts when embarking on a mathematical proof.

An important aspect of intelligent behaviour as studied in AI is goal-based problem solving, a framework in which the solution of a problem can be described by finding a
sequence of actions that lead to a desirable goal. A goal-seeking system is supposed to be
cconnected to its outside environment by sensory channels through which it receives
information about the environment and motor channels through which it acts on the
environment. (The term "afferent" is used to describe "inward" sensory flows, and "efferent"
is used to describe "outward" motor commands.) In addition, the system has some means of
storing in a memory information about the state of the environment (afferent information)
and information about actions (efferent information).

Ability to attain goals depends on building up associations, simple or complex,
between particular changes in states and particular actions that will bring these changes
about. Search is the process of discovery and assembly of sequences of actions that will lead
from a given state to a desired state. While this strategy may be appropriate for machine
learning and problem solving, it is not always suggested for humans (e.g. cognitive load
theory and its implications).

Note that, in order for MEA to be effective, the goal-seeking system must have a
means of associating to any kind of detectable difference those actions that are relevant to
reducing that difference. It must also have means for detecting the progress it is making (the
changes in the differences between the actual and the desired state), as some attempted
sequences of actions may fail and, hence, some alternate sequences may be tried.

When knowledge is available concerning the importance of differences, the most
important difference is selected first to further improve the average performance of MEA
over other brute-force search strategies. However, even without the ordering of differences
according to importance, MEA improves over other search heuristics (again in the average
case) by focusing the problem solving on the actual differences between the current state and
that of the goal.

3.2.5 Penalty Co–efficient Method

Penalty Co-efficient methods are a certain class of algorithms for solving constrained
optimization problems. A penaltyco-efficient method replaces a constrained optimization
problem by a series of unconstrained problems whose solutions ideally converge to the
solution of the original constrained problem. The unconstrained problems are formed by
adding a term, called a penalty co-efficient function, to the objective function that consists of
a penalty co-efficient parameter multiplied by a measure of violation of the constraints. The
measure of violation is nonzero when the constraints are violated and is zero in the region where constraints are not violated.

Penalty functions have been a part of the literature on constrained optimization for decades. Two basic types of penalty functions exist; exterior penalty functions, which penalize infeasible solutions, and interior penalty functions, which penalize feasible solutions. The main idea of interior penalty functions is that an optimal solution requires that a constraint be active (i.e., tight) so that this optimal solutions lies on the boundary between feasibility and infeasibility. Knowing this, a penalty is applied to feasible solutions when the constraint is not active, so-called “interior solutions.” For a single constraint, this approach is straightforward (although it has not been seen in the evolutionary computation literature), however for the more common case of multiple constraints, the implementation of interior penalty functions is considerably more complex.

Three degrees of exterior penalty functions exist: (1) barrier methods in which no infeasible solution is considered, (2) partial penalty functions in which a penalty is applied near the feasibility boundary, and (3) global penalty functions that are applied throughout the infeasible region. This method employs a “flexible” penalty function to promote convergence, where during each iteration the penalty parameter can be chosen as any number within a prescribed interval, rather than a fixed value. This increased flexibility in the step acceptance procedure is designed to promote long productive steps for fast convergence.

3.3 Development of Visual C++ Programming

Microsoft Visual C++ (often abbreviated as MSVC or VC++) is a commercial, Integrated Development Environment (IDE) product from Microsoft for the C, C++, and C++/CLI programming languages. It features tools for developing and debugging C++ code, especially code written for the Microsoft Windows API (Application Programme Interface), the DirectXAPI, and the Microsoft .NET Framework.

Many applications require redistributable Visual C++ packages to function correctly. These packages are often installed independently of applications, allowing multiple applications to make use of the package while only having to install it once. These Visual C++ redistributable and runtime packages are mostly installed for standard libraries that many applications use.
3.3.1 Features of VC++

It has the tools for coding and debugging of the visual codes. The main features of this VC++ are as follows:

1) Smart Pointer
2) New Containers
3) Expression Parsing
4) Polymorphism
5) Exception Handling
6) Garbage collection
7) Type traits and sophisticated random number generators

It features tools for developing and debugging C++ code, especially code written for the Microsoft Windows API, the DirectX API, and the Microsoft .NET Framework. Stack based buffer over run protects the program from executing malicious code. Function epilogue code checks the returned cookie for similarity, if there is a mismatch it stops executing the code. It also moves around the stack which makes the data corruption harder. These features are present in Visual C++ 2005. It also mitigates various buffer over runs through memory usage reduction.

Some of the new features which are introduced to VC++ are the large set of new MFC classes which will certainly help in building the modern user interfaces. TR1 is the major addition to the package of VC++. This also has new user interface, adjustable panes, and support for Microsoft office ribbon, controls, dialog boxes and windows.

Frame describes about the borders, dimensions and location of the window. Two types of MFC applications are present which uses in a frame. Application based frames uses a concept known as Document/View architecture. This architecture allows a certain program to be present in applications.

Functionality of VC++ defines the huge number of templates which can help to construct a function object which defines the operators. Function pointer can be described by a function object; it stores the information which can be accessed during the functional call.

The Microsoft Visual C++ includes the tools listed below.
**Microsoft Foundation Classes (MFC)**

MFC is a large and extensive C++ class, hierarchy library that makes the Windows application development easier. CFrameWnd is one of those very important class and it is used frequently. User defined classes can be created by CFrameWnd.

**App Wizard**

App Wizard is a code generator that creates a working skeleton of a Windows application with features, class names, and source code filenames. It helps the programmer to start quickly with a new application.

**Class Wizard**

Class Wizard is a program which generates code for a new class or a new function. It writes the prototypes, function bodies, and code to connect the messages to the application framework.

**App Studio**

App Studio is a resource editor which includes WYSIWYG menu editor and a powerful dialog box editor.

**Windows Programming**

In Windows application you need a package to your application with GUI elements and windows. This is to provide consistent UI response for all windows applications to avoid the relearn about the same activities for user.

To do the same in Windows it requires a Windows API is to be present. This API library makes it easier for the developer to use the graphical resources because they are function calls. In fact there is a chance to get the source code for these libraries so that they can exploit for your developments if necessary. While programming in Visual Basic (VB), the Windows API did not used directly.VB is another layer built on top of the Windows API that makes it still easier for the developer. When we placed a command button on the form, VB would call several API functions to get the job done. Even a simple application created with standard API function calls is a nightmare to develop.
In Visual C++, Microsoft provides a Class Library called the Microsoft Foundation Class (MFC). These are set of OOP tools for Win32 applications. This is once again a layer between the developer and the API, and provides a powerful toolkit for OOP for windows.

The Microsoft Foundation Class Library (MFC) encapsulates, or "wraps," much of (but not all of) the Win32 API. MFC versions 2.x and earlier encapsulated the 16-bit Windows API. In general, MFC supplies classes representing key Windows objects, such as windows, dialog boxes, brushes, pens, and fonts. The member functions of these classes wrap most of the important Win32 API functions associated with the encapsulated object. That is, the MFC class member function calls the Win32 API function (it may do other things as well).

The MFC has some key advantages

- Complete support for all windows functions,
- Same names as in Windows API for same task,
- Message map macros. All messages are mapped to member functions,
- Extensive exception handling,
- Small code and almost as fast as a C implementation &
- Support for COM (Component Object model) Objects

C++ is a programming language and Visual C++ is an IDE (Integrated Development Environment) for developing with languages such as C and C++.

VC++ contains tools for, amongst others, developing against the .net framework and the Windows API.

Advantages of VC++ over other programming languages:

1. Compatible with C source code, except for a few corner cases.
2. Write Once, Compile Anywhere (WOCA).
3. Allows procedural programming, functional programming, object-oriented programming, generic programming, and template meta-programming. Favours a mix of paradigms.
4. Runs as native executable machine code for the target instruction set(s).
5. Has multiple binary compatibility standards (commonly Microsoft (for MSVC compiler) and Itanium/GNU (for virtually all other compilers)).
6. Supports native unsigned arithmetic.
7. Standardized minimum limits for all numerical types, but the actual sizes are implementation-defined. Standardized types are available through the standard library `<cstdint>`.

8. Memory management can be done manually through `new / delete`, automatically by scope, or by smart pointers. Supports deterministic destruction of objects. Garbage collection ABI standardized in C++11, though compilers are not required to implement garbage collection.

9. Resource management can be done manually or by automatic lifetime-based resource management (RAII).

10. Allows explicitly overriding types as well as some implicit narrowing conversions (for compatibility with C).

11. The C++ Standard Library was designed to have a limited scope and functionality but includes language support, diagnostics, general utilities, strings, locales, containers, algorithms, iterators, numeric, input/output, random number generators, regular expression parsing, threading facilities, type traits (for static type introspection) and Standard C Library. The Boost library offers more functionality including network I/O.

12. Source code can be written to be platform-independent (can be compiled for Windows, BSD, Linux, Mac OS X, Solaris, etc., without modification) and written to take advantage of platform-specific features. Typically compiled into native machine code, must be re-compiled for each target platform.

### 3.4 Other programming languages

i) **C language**

C is an imperative (procedural) language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal runtime support. C was therefore useful for many applications that had formerly been coded in assembly language, such as in system programming.

Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant and portably written C program can be compiled for a very wide variety of computer platforms and operating systems with few
changes to its source code. The language has become available on a very wide range of platforms, from embedded microcontrollers to supercomputers.

**ii) C++**

C++ is a statically typed, free-form, multi-paradigm, compiled, general-purpose programming language. It is regarded as an intermediate-level language, as it comprises a combination of both high-level and low-level language features. It adds object oriented features, such as classes, and other enhancements to the C programming language. Originally named C with Classes, the language was renamed C++ in 1983. C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms.

**iii) JAVA**

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "Write Once, Run Anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java support all the features of object oriented programming language such as Abstraction, Encapsulation, Inheritance, Polymorphism and Dynamic binding etc. Java is one of the most popular programming languages in use, particularly for client-server web applications.

**vi) Object Oriented Programming:**

OOP stands for Object Oriented Programming and the language that support this OOP features is called Object Oriented Programming Language. An example of a language that support this Object oriented features is C++. The Objects Oriented programming language supports all the features of normal programming languages. In addition it supports some important concepts and terminology which has made it popular among programming methodology.
The important features of Object oriented programming are:

- Data Hiding
- Encapsulation
- Inheritance
- Reusability
- Polymorphism & Overloading

3.5 MATLAB Software

MATLAB is a powerful computing system for handling the calculations involved in scientific and engineering problems. The name MATLAB stands for Matrix Laboratory, because the system was designed to make matrix computations particularly easy. The MATLAB software package is used for computation in engineering, science, and applied mathematics.

MATLAB is usually not the tool of choice for high-performance computing. Whatever you think of these or other limitations of MATLAB, they have not held back its popularity. Rapid code development and interaction with data often trump execution speed, and the integrated graphics and expert routines that come with MATLAB can be decisively helpful. Even for speed hungry users, MATLAB can be a valuable environment in which to explore and fine-tune algorithms before creating production code in another environment. Successful computing languages and environments reflect a distinctive set of values. In MATLAB, those values include an emphasis on experimentation and interaction with data and algorithms; syntax that is compact, friendly, and interactive (rather than tightly constrained and verbose).

MATLAB was first adopted by control design engineers but quickly spread to many other domains. It is popular in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved with image processing.
3.5.1 GUI in MATLAB

GUIs (Graphical User Interface) make things simple for the end-users of the program. If GUIs were not used, people would have to work from the command line interface, which can be extremely difficult and frustrating. Imagine if you had to input text commands to operate your web browser. It wouldn’t be very practical.

![Fig 3.1 A simple GUI](image)

Creating GUI with GUIDE in MATLAB

GUIDE (Graphical User Interface Development Environment) is a tool for laying out and programming GUI’s. A GUI uses graphics and text input to make using MATLAB much more user friendly for people who are unfamiliar with it. GUI’s can be created without using GUIDE but laying out the design of the window can be very time consuming. To open GUIDE click on the start button in the bottom left corner of MATLAB and select START→MATLAB→GUIDE.

**Initializing Guide**

**Step 1:** First, open up MATLAB. Go to the command window 1. And type in guide

![Fig 3.2 Initializing GUIDE](image)
It is similar to writing a command statement that creates a variable, or perform an addition operation or to create a small variable in MATLAB. The output is also shown in command window itself.

You just have to type the command GUIDE in command window for initializing GUI.

**Step 2:** Choose the first option Blank GUI (Default). MATLAB

![GUIDE Quick Start](image1)

**Fig 3.3 Starting GUI with GUIDE**

**Step 3:** you will see the following screen

![myAdder.fig](image2)

**Fig 3.4 Blank GUI**
Step 4: design GUI as you required.

![Designing GUI](image1)

**Fig 3.5** Designing GUI

Step 5: Create visual aspects using property inspector

![Property inspector](image2)

**Fig 3.6** Property inspector

3.5.2 Programming with GUI in MATLAB using call back

A call back is a functions executed whenever the user initiates an action by clicking for example on a button or pressing a key on the keyboard. Programming a call back is therefore the most important part of writing a GUI. For example in the GUI illustrated above, we would need to program a call back for the button Add. This is provided as a call back in the code. The code is provided below and illustrates how to write a simple call back.

```matlab
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Example code for button press

% For example, you could update the display or change the state of other objects
% handles.myValue = handles.myValue + 1; % Increment a value
% GUIDATA(hObject, eventdata, handles); % Update the GUI
```

% object handle to pushbutton1 (see GCBO)
In this piece of code we can get the numbers as strings from the edit boxes and then convert them into numbers using the `str2double` function provided in MATLAB. I then set the string for the other edit box as the sum of these two numbers. This completes the simple example of a GUI needed to add two numbers. To illustrate a more complex example I show how a simple exponential function can be plotted and you change the function's parameters, with a little bit of imagination you could make it plot any arbitrary function you enter. To make the example even more complex I have two GUlS, one is the control GUI and the other is the plotting GUI, this allows the user to program some of the more complicated functionality expected out of the modern GUI systems.

### 3.6 Summary

The applications of AI search algorithms in the field of manufacturing are explained by considering the process planning and scheduling problems. The reasons behind the selection of MEA and PCM algorithm are given indirectly by explaining all search algorithms of AI techniques.

The Visual C++ is selected for the software development of present application as it seems to be more applicable. The VC++ features, tools, classes and advantages of VC++ over the other programming languages are explained. Moreover the software development in MATLAB package is given in detail with required information.