CHAPTER 1

INTRODUCTION

1.1 PRODUCTION PLANNING AND CONTROL

Production Planning and Control (PPC) system is outlined best by considering the information flow in a manufacturing system. Pinedo (1995) expresses a simplified sketch of information flow, by neglecting the interfaces to other functions of a manufacturing environment is shown in figure 1.1 [1]. Demand forecasts and customer orders are input to the medium to long term production planning. A master schedule is built resulting in the demand of end product quantities and their desired due dates. On the basis of quantities and due dates the material required for production is planned according to the volume and the period. This process results in material requirements of forthcoming production periods which have to be supplied in time. The material requirements planning are highly interwoven with the capacity planning. Here, temporal assignments of orders to the available processing capacity are shifted such that capacity bottlenecks are avoided and due dates are kept. Up to this stage coarse grained production planning is performed on the basis of customer orders. Now shop orders (jobs) and their release times are introduced as an outcome of the capacity planning.
Figure 1.1 Information flow in a Manufacturing system
The jobs are input to the scheduling engine of the PPC system. Scheduling performs allocation by keeping capacity constraints and finally produces a detailed schedule, i.e. determines the periods of processing some job on its dedicated machines. Thereby scheduling pursues an economically motivated objective. Typically, a reduction of the work in-process inventory is pursued by increasing the throughput of jobs. Moreover, scheduling aims to avoid delivery delays of customer orders and tries to make full use of the available production capacity. Production planning is completed with dispatching already scheduled jobs to the shop floor management.

Scheer (1989) indicates that scheduling function is embedded in the domain of production planning and control (PPC) [2]. The purposes covered by a scheduling allocate resources over time in order to perform a number of tasks. Typically resources are limited and therefore tasks are assigned to resources in a temporal order. From an economic point of view limited resources are scarce goods and consequently the problem of task scheduling is more important than just academic relevance. Following Van Dyke Parunak (1992), scheduling is circumscribed by asking what has to be done where and when [3]. A task (what) occupies a dedicated resource exclusively (where) for some period of time (when). A group of task primitives may form a complex, in which several tasks have to use resources in a certain order. In this way, the temporal order of resource allocations is restricted by dependencies among the task primitives. Any process that defines a subset of what X where X when can be said to execute scheduling.

Scheduling theory is concerned with the optimal allocation of the scarce resources to activities over time. The practice of this field dates to the first time two humans contended for a shared resource and developed a plan to share it without bloodshed. The theory of the design of the algorithms for scheduling is still younger, but has significant histories [4–14].

Scheduling problems arise in a variety of settings and are illustrated by the following examples:
Example 1:

Consider the central processing unit of a computer that must process a sequence of jobs that arrive over time. In what order, should the jobs be processed in order to minimize the average time that a job in the system from arrival to the completion?

Example 2:

Consider a team of five astronauts preparing for the reentry of their space shuttle into the atmosphere. There is a set of tasks that should be accomplished by the team before reentry. Each task must be carried out by exactly one astronaut, and certain task can not be started until other tasks are completed. Which jobs should be performed by which astronaut and in which order, to ensure that the entire set of tasks is accomplished as quickly as possible?

Example 3:

Consider a factory that produces different sorts of widgets. Each widget must be processed by Machine 1 then Machine 2 and then Machine 3, but different widget requires different amounts of processing time on different machines. The factory has orders for a batch of widgets; each order has a date by which it must be completed. In what order should the machines work on different widgets in order to ensure that the factory completes as many orders as possible on time?

More generally, scheduling problems involve tasks that must schedule on facilities subjected to certain constraints to optimize some cost function. The goal is to specify a schedule that specifies on which facility each task is to be executed.
There are different classes of scheduling problems and are broadly classified as:

- Machine scheduling
- Project scheduling
- Material handling scheduling (Automated Guided Vehicle scheduling, Robotics scheduling etc.)
- University Time tabling

This thesis addresses about machine scheduling problems.

1.2 MACHINE SCHEDULING

In manufacturing systems, operations (tasks) are processed by machines (resources) for a certain processing time (time period). Typically, the number of machines available is limited and a machine can process a single operation at a time. Often, the operations cannot be processed in arbitrary orders but must obey a prescribed processing order. Jobs often follow technological constraints which define a certain category of shop floor like:

- Single machine shop
- Flow shop
- Job shop
- Open shop

In a single machine shop, a single machine or facility is available to perform tasks on different jobs. In a job shop technological constraints may differ from job to job. In a flow shop all jobs pass the machines in an identical order. In an open shop no technological restriction exists and therefore the operations of jobs may be processed in arbitrary orders.

Machine scheduling determines the starting times of operations without violating technological constraints such that processing times of identical machines do not overlap in time. The resulting time table is called as a schedule. Most of the machine scheduling
problems is classified as deterministic models and stochastic models. In deterministic scheduling models, a set of jobs has to be processed by a set of machines and certain performance measures or objectives have to be optimized. In stochastic scheduling, the input job data follows a probabilistic distribution. Typical objectives are the reduction of the make span of an entire production program, the minimization of mean job tardiness, the maximization of machine load or some weighted average of many similar criteria. Since the number of different problem types that have been considered in the area of machine scheduling is enormous, it is convenient to use the standard classification scheme proposed by Graham et al. (1979) [15]. The scheduling problems can be represented by a triplet notation \( a/\beta/\gamma \).

1.2.1 \( a/\beta/\gamma \)-Notation

A problem is referred to in the three-field notation \( a/\beta/\gamma \), with the following intended meaning:

- The field \( \alpha \) specifies the machine environment. For instance, \( \alpha = J \) denotes the job shop model and \( \alpha = 1 \) is used for single machine shop.

- The field \( \beta \) contains the job characteristics. It can be empty, which implies the default of non-preemptive, independent jobs. Possible entries are, among many others, \( r_j \) if release dates are present, \( prec \) for precedence constrained jobs, or \( pmtn \) for preemptive jobs.

- The field \( \gamma \) denotes the objective function. It is generally a function of the completion times of the jobs. For the total weighted tardiness \( \gamma = \sum w_j T_j \) and for the makespan \( \gamma = C_{\text{max}} \).

As an example, \( 3 \mid r_j, \text{prec} \mid \sum w_j C_j \) is the problem to minimize the total weighted completion time of precedence constrained jobs with release dates on three parallel, identical machines. Whenever to specify only the machine and job characteristics, but not a particular objective function, a notation \( a/\beta \ast \) is used.
Generalizations of this three-field notation have been suggested, particularly in order to capture also resource constrained project scheduling problems and other scheduling models that cannot be represented in the original three-field notation of Graham et al. [15]. Currently, however, none of these generalizations seems to prevail. Further details on the classification of scheduling problems are available in [16 – 18].

In this thesis, the deterministic single machine scheduling problem, classified as $(l \mid \sum w_j T_j)$ with the objective of minimization of total weighted tardiness is considered.

1.3 SINGLE MACHINE SCHEDULING

The single machine scheduling problem aims in sequencing a set of jobs for processing on a single processor or machine, usually bottleneck in the system. In many applications, the bottleneck machine can be seen as an input/output bay, where the orders arrive to be further fulfilled or where the completed orders are gathered before shipped to the customer [19, 20]. So, the study of single machine scheduling problems is very important for several reasons, probably the most relevant of which is that good solutions to this problem provide a support to manage and model the behavior of more complex systems. In these systems, it is important to understand the working of their components, and quite often the single machine problem appears as an elementary component in a larger scheduling problem [12].

The basic single machine problem considered in this thesis is characterized by the following conditions:

- a set of $n$ independent jobs is available for processing at time zero and the job descriptors are known in advance
- a machine is continuously available and is never kept idle
- while the set up times for the jobs are independent of job sequence and can be included in processing times
- jobs are processed to completion without preemption
Under these conditions there is a one-to-one correspondence between a sequence of these \( n \) jobs and a permutation of the job indices. The total number of different solutions to the single machine scheduling problem is \( n! \).

Furthermore, due to the complexity studies conducted during the last four decades, it is now widely understood that most practical single machine scheduling problems are Non Polynomial (NP) hard. That is, the time the best possible algorithm will need to solve the problem increases in the worst case exponentially with the size of the problem. For reasonably sized problems, the computational time needed for solving the problem can be from very long up to intractable. In general, since most of the scheduling problems are NP hard, there are no known algorithms guaranteed to give an optimal solution and run in polynomial time. This tends to rely on the use of heuristic search algorithms.

1.4 HEURISTIC SEARCH ALGORITHMS

Heuristic search algorithms are often developed and used to solve many difficult NP-hard type computational problems in science and engineering. Heuristics can derive near optimal solutions in considerably less time than the exact algorithms. Heuristics often seek to exploit special structures in a problem to generate good solutions quickly. However, there is no guarantee that heuristics will find an optimal solution.

Heuristics are obtained by
- using a certain amount of repeated trials,
- employing one or more agents viz. neurons, particles, chromosomes, ants, and so on,
- operating with a mechanism of competition and cooperation,
- embedding procedures of self modification of the heuristic parameters or of the problem representation.

Heuristic search algorithms utilize the strengths of individual heuristics and offer a guided way for using various heuristics in solving a difficult computational problem.
According to Osman (1996), a heuristic search "is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces..." [21, 22]. Heuristic search algorithms have shown promise for solving "...complex combinatorial problems for which optimization methods have failed to be effective and efficient." Common methods include:

- Tabu search [23 – 26],
- simulated annealing [27, 28],
- greedy random adaptive search procedures (GRASP) [29,30],
- iterated local search [31],
- genetic algorithms [32], and
- ant colony optimization [33].

There are different ways to classify and describe heuristic search algorithms which are explained in the subsequent sections:

1.4.1 Classification of Heuristic Search Algorithms

Depending upon the characteristics to differentiate between search algorithms, several classifications are possible and each of them being the results of a specific view point. The most important methods of classification are:

- Nature inspired vs Non nature inspired
- Population based vs Single point search
- Dynamic vs Static objective function
- One vs Various neighborhood structure
- Memory Usage vs Memory less method
Nature inspired vs Non nature inspired

Perhaps, the most intuitive way of classifying heuristic search algorithms is based on the origin of the algorithms. There are nature inspired algorithms like evolutionary algorithms and ant algorithms and non nature inspired algorithms like Tabu search and iterated local search / improvement algorithms. This classification is not meaningful for the following two reasons. First, many hybrid algorithms do not fit in either class or in a sense that it fit both at the same time. Second, sometimes it is difficult to clearly tell the genesis of an algorithm.

Population based vs Single point search

Another characteristic which can be used for the classifications is the way of performing the search. Does the algorithm work on a population or on a single solution at a time? Algorithms working on single solution are called as trajectory methods and encompass local search based heuristics. They all share the property of describing a trajectory in the search space during the search process. Population based methods on the contrary perform search process which describe the evolution of a set of points in the solution space.

Dynamic vs Static objective function

Search algorithms can also be classified according to the way they make use of the objective function. While some algorithms keep the objective function given in the problem representation “as it is” and some others like guided local search will modify during the search. The idea behind this search is to escape from the local optima by modifying the search landscape. Accordingly, during the search the objective function is altered by trying to incorporate information collected during the search process.
One vs Various neighborhood structure

Most search algorithms work on single neighborhood structure. In other words, the fitness landscape which is searched doesn’t change in the course of the algorithm. Other algorithms use a set of neighborhood structures which gives the possibility to diversify the search and tackle the problem jumping between different landscapes.

Memory Usage vs Memory less method

A very important feature to classify the heuristic search algorithms is whether they use memory of search history or not. Memory less algorithms perform a Markov process, as the information they need is only the current state of the search process. There are several different ways of making use of memory. Usually it will be differentiated between short term and long term memory structures. The first usually keeps track of recently performed moves, visited solutions or, in general, decisions taken. The second is usually the accumulation of synthetic parameters and indexes about the search. The use of memory is nowadays recognized as one of the fundamental elements of the powerful heuristics.

The performance of any search algorithm depends on the structure of heuristic search involved.

1.5 EVOLUTIONARY HEURISTIC SEARCH

Since uninformed search by enumeration methods seems computational prohibitive for large search spaces, heuristic search receives increasing attention [7]. Instead of searching the problem space exhaustively, Reeves (1993) informs that modern heuristic techniques concentrate on guiding the search towards promising regions of the search space [34].

A wide range of different heuristic search techniques have been proposed which all have some basic component parts in common. They are:
A representation of partial and complete solutions is required.

Operators are needed which either extend partial solutions or modify complete solutions.

An objective function is needed which either estimates the costs of partial solutions or determines the costs of complete solutions.

The most crucial component of heuristic search techniques is the control structure which guides the search.

Finally, a condition for terminating the iterative search process is required.

Prominent heuristic search techniques are, among others, simulated annealing, Tabu search and evolutionary algorithms. The first two of them have been developed and tested extensively in combinatorial optimization. To the contrary, evolutionary algorithms have their origin in continuous optimization. Nevertheless, the components of evolutionary algorithms have their counterparts to other heuristic search techniques. A solution is called an individual which is modified by operators like crossover and mutation. The objective function corresponds to the fitness evaluation. The control structure has its counterpart in the selection scheme of evolutionary algorithms.

In evolutionary algorithms, the search is loosely guided by a multi-set of solutions called a population, which is maintained in parallel. After a number of iterations (generations) the search is terminated by means of some criterion. A careful evaluation of the suitability of evolutionary algorithms for machine scheduling is subject of this thesis. Thereby particular attention is paid to the conditions which must be fulfilled so that guiding the search succeeds.

The main aim of this thesis is to solve the problems of higher sizes within reasonable time. In this thesis, three different heuristic search algorithms are formulated and used to solve the single machine scheduling problems with objective of minimizing the total weighted tardiness.
They are:

• Heuristic Improvement algorithm - trajectory method operating on a single sequence
• Iterated Local Improvement Evolutionary Algorithm - hybridized search algorithm with size of population as two.
• Self Improving Mutation Evolutionary Algorithms - hybridized search algorithm with size of population greater than two.

1.6 ORGANIZATION OF THESIS

The organization of this thesis is as follows:

Chapter 1 provides an introduction for scheduling problems and heuristic search algorithms. It indicates when scheduling can be of practical interest, explains the different heuristic approaches applied to the single machine scheduling problems.

Chapter 2 explains the background and literature review of the current trends in single machine scheduling and the different literature related to the various techniques used to solve scheduling problems. It cites different works which are carried out in the field of single machine scheduling problems and also provides an insight to research works of the various heuristic search algorithms.

Chapter 3 describes briefly about the single machine scheduling problems. The mathematical representation of single machine total weighted tardiness problem and single machine total tardiness problem are provided in this chapter.

Chapter 4 gives the different method of generating the test problem instances of single machine scheduling problems and provides an introduction to the single machine total weighted tardiness benchmark problem instances available in the Operations Research (OR) library. It also indicates the best known results for the benchmark instances.
reported so far. The details of the single machine total tardiness problem and the unsolved problems within the time limits are also given in this chapter.

Chapter 5 deals with the development of a heuristic improvement technique to solve benchmark instances of single machine total weighed tardiness problem.

Chapter 6 proposes an iterated local improvement evolutionary algorithm (ILIEA) to get good results in reasonable time. An evolutionary perturbation tool is introduced in the crossover operation and a self improvement technique is adopted in the mutation part of this proposed iterated local improvement evolutionary algorithm. The performance of this algorithm is studied and compared with the best known values of the benchmark instances of single machine total weighted tardiness problem.

Chapter 7 introduces self improving mutation evolutionary algorithm (SIMEA) with more number of individuals in the population. Two versions of SIMEA are tested on the benchmark instances of single machine total weighted tardiness problems and the best performing version is used to solve the problems of higher sizes.

Chapter 8 gives the performance comparison of the different algorithms proposed in this thesis.

Chapter 9 provides some concluding remarks and suggests the future scopes of this research work.