

Chapter 1

INTRODUCTION

Just-in-Time (JIT) is defined as “a philosophy of manufacturing based on planned elimination of all waste and on continuous improvement of productivity”. It also has been described as an approach with the objective of producing of the right product in the right place at the right time. Waste results from any activity that add cost without adding value, such as the unnecessary moving of materials, the accumulation of excess inventory, or the use of faulty production methods that create products requiring subsequent rework. JIT should improve profits and return on investment by reducing inventory levels (increasing the inventory turn over rate), reducing variability, improving product quality, reducing other costs (such as those associated with machine setup and equipment breakdown). In a JIT system, underutilized (excess) capacity is used instead of buffering inventories to hedge against problems that may arise.

The primary characteristics differentiating JIT system from conventional systems is that subsequent processes within the manufacturing system “pull” their part requirements from the preceding processes. This pull process results in the production of only the required parts in the required quantities at the required time, Miltenburg et al [21] and Monden [23]. Necessarily, the final assembly process becomes the primary focus for control in JIT manufacturing. The most important goal for a JIT production system is to ensure that the quantity of each part used by the assembly process is kept as close to constant as possible per unit time, Monden [23]. Determining the sequence of final assembly which achieves this goal is commonly referred to in the literature as leveling or balancing the schedule, Hall [9]. A balance schedule minimizes the variability in the production rate of all parts and products within the JIT system. This reduces the possibility of “shock waves” caused by sudden increases in part requirements, which may result in shortages, or by sudden decreases, which may create excessive inventories.

Determining a balanced schedule is the corner stone of the Toyota production system, Monden [23]. To construct such a schedule, Toyota has employed a heuristic procedure known as Goal Chasing Method. For most industrial applications, however, determining an optimal balanced schedule is a very difficult combinatorial problem, Kubaik [15] and Kubaik et al [17].

A considerable amount of research into balanced schedule problems has been undertaken during the past decade. Although much of the interest into JIT manufacturing was spurred on by the description of the Toyota production system Monden [23], the catalyst for balanced schedule research was the seminal work of Miltenburg [21]. Miltenburg [21] transformed Toyota's balanced schedule problem into a nonlinear integer programming problem. The objective of this formulation was to determine the sequence of final assembly which minimize the 'sum of deviations' (min-sum) of actual production from the desired quantity of production. This original model as subsequently extended to multi-level assembly systems where the implied part demands for the outputs from feeder processes were also considered when fixing the sequence of final products assembly, Miltenburg et al [21] [20]. Beside this, several heuristics procedures have been proposed for solving balanced schedule, Inman and Bulfin [11], and Jost [12].

A dynamic programming algorithm for optimizing the single-level (i.e. product level), min-sum problem was provided by Miltenburg et al [22]. Kubaik and Sethi [16] demonstrated that the objective function can be represented by penalties for deviation from the most even but unrealizable (i.e. fractional) distribution of demand and that if these penalty functions are non-negative and convex, then this problem can be reduced to an assignment problem. Steiner & Yeomans [25] formulated an efficient graph-theoretic optimization algorithm for minimizing the maximum (min-max) absolute deviation of actual production from the desired quantity of production. A procedure for generating several Pareto optimal solutions to the problem combining both the min-sum and min-max objectives is described in Steiner & Yeomans [25]. Kubaik [15] and Kubaik et al. [17] proved

that balanced schedule problems with two or more production levels are NP-hard. Optimization algorithms for both min-sum and min-max multi-levels problems appear in Kubaik et al. [17]. Beside this, Miltenburg [21] considers the quantity of each part used by mixed model assembly line per unit time should be kept as constant as possible. Beside this, an efficient algorithm for obtaining an optimal solution for maximum absolute-deviation objective in single level with chain constrained had been developed by Dhamala [5]. Kovalyov, Kubiak, and Yeomans [13], have observed the computational complexity of balanced JIT optimization algorithm and showed that most of single-level JIT problems could be efficiently solvable. Similar analytical study had been done by Dhamala and Khadka [6].

This dissertation has been organized as follows. Chapter 2 explains some fundamental concepts of Computational Complexity Theory, Complexity Classes and Scheduling and its related problems. Chapter 3 briefly describes the mixed model production system. It also studied different mathematical models for mixed model JIT problems for single and multi-levels problems.

Chapter 4 explores the solution procedure for min-sum PRVP. Different algorithms for the solution are described in this chapter. The EDD Algorithm to find optimal solution for min-sum problem is reviewed. The nearest integer point method to find a solution for min-sum problem is included. Cost assignment approaches for min-sum PRVP is described. Dynamic Programming approach to solve the min-sum problem, which is found fruitful for large size problem, is illustrated. Moreover, an algorithm so called min-sum absolute chain algorithm with extended EDD is also presented. Chapter 5 considers the combination of different non-overlapping chains to find out all possible sequences from the input chain sequences. Moreover, an attempt to find the best sequence among them is done. Chapter 6 has concluded this study with some remarkable achievements like combining different non-overlapping chains and explores optimal sequence with minimum cost value in mixed-model JIT production system.

Chapter 2

FUNDAMENTAL CONCEPTS

2.1 Turing Machine

Turing Machine is basic abstract symbol-manipulating device which, despite their simplicity, can be adapted to simulate the logic of any computer algorithm. It was described in 1936 by Alan Turing. Turing Machine is not intended as a practical computing technology, but a thought experiment about the limits of mechanical computation. Thus, it was not actually constructed. Studying its abstract properties yields many insights into computer science and complexity theory.

A Turing Machine that is able to simulate any other Turing Machine is called a Universal Turing Machine (UTM, or simply a universal machine). A more mathematically-oriented definition with a similar "universal" nature was introduced by Alonzo Church, whose work on lambda calculus intertwined with Turing's in a formal theory of computation known as the Church-Turing thesis. The thesis states that Turing machines indeed capture the informal notion of effective method in logic and mathematics, and provide a precise definition of an algorithm or 'mechanical procedure'. Some of the examples of Turing Machine are: Turing's very first machine, Copy routine, 3-state busy beaver, etc.

2.2 Computational Complexity

Computational complexity theory is a branch of the theory of computation in computer science that investigates the problems related to the resources required to run algorithms, and the inherent difficulty in providing algorithms that are efficient algorithms for both general and specific computational problems. Complexity theory attempts to describe how difficult it is for an algorithm to find a solution to a problem. This differs from computability theory, which describes whether a problem can be solved at all. Furthermore, much of complexity theory

deals with decision problems. A decision problem is one where the answer is always "yes" or "no". Some problems are undecidable, or at least seem so, so complexity theory can be used to distinguish problems where it is certain to get a correct "yes" or "no" (not necessarily both). A problem that reverses which can be relied upon is called a complement of that problem. Complexity theory analyzes the difficulty of computational problems in terms of many different computational resources. A problem can be described in terms of many requirements it makes on resources: time, space, randomness, alternation, and other less-intuitive measures (vague).

2.3 Function

Functions play a fundamental role in all areas of mathematics, as well as in other Sciences, Information Communication Technology and Engineering. However, the intuition pertaining to functions, notation, and even the very meaning of the term "function" varies among the fields. More abstract areas of mathematics, such as set theory, consider very general types of functions that may not be specified by a concrete rule or be governed by familiar principles. In the most abstract sense, the distinguishing feature of a function is that it relates exactly one output to each of its admissible inputs. Such functions need not involve numbers. For example, a function might associate each member of a set of words with its own first letter.

Given two sets A and B , a function f is a binary relation on $A \times B$ such that for all $a \in A$, there exists precisely $b \in B$ such that $(a, b) \in f$. The set A is called domain of f , and the set B is called co-domain of f . We write $f : A \rightarrow B$ and if $(a, b) \in f$, we write $b = f(a)$, since b is uniquely determined by choice of a . Two functions f and g are equal if they have the same domain and co-domain and if, for all a in the domain, $f(a) = g(a)$. A finite sequence of length n is a function f whose domain is the set of n integers $\{0, 1, 2, \dots, n-1\}$. Finite sequence is denoted by listing its values: $\{f(0), f(1), f(2), \dots, f(n-1)\}$. An infinite sequence is a function whose

domain is set of \mathbb{N} natural numbers. For example, the Fibonacci sequence, defined by recurrence, is the infinite sequence $\{0, 1, 1, 2, 3, 5, 8, \dots\}$.

2.4 Complexity Classes

In computational complexity theory, a complexity class is a set of problems of related complexity. A typical complexity class has a definition of the form: the set of problems that can be solved by abstract machine M using $O(f(n))$ of resource R (n is the size of the input). A complexity class is the set of all the computational problems which can be solved using a certain amount of a certain computational resources. There are several complexity classes in the theory of computation. Some of the major classes are discussed below.

2.4.1 Class P

The complexity class P is the class of decision problems that can be solved by a deterministic machine in polynomial time. This class corresponds to an intuitive idea of the problems which can be effectively solved in the worst cases.

Example 2.1 The problem of sorting n numbers can be done in $O(n^2)$ time using the quick sort algorithm in worst case. Thus all sorting problems are in P .

2.4.2 Class NP

The complexity class NP is the set of decision problems that can be solved by a non-deterministic Turing machine in polynomial time. This class contains many problems that people would like to be able to solve effectively, including the Boolean satisfiability problem, the Hamiltonian path problem and the vertex cover problem. All the problems in this class have the property that their solutions can be checked efficiently.

Example 2.2 A vertex cover of an undirected graph $G = (V, E)$ is a subset of $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ and $v \in V'$ or both. That is, each edge

touches at least one vertex V' . The vertex-cover problem is to find such a vertex cover of minimal cardinality. This problem is in NP.

2.4.3 NP-Complete

In computational complexity theory, the complexity class NP-complete (abbreviated NP-C or NPC, NP standing for Nondeterministic Polynomial time) is a class of problems having two properties:

- Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP.
- If the problem can be solved quickly (in polynomial time), then so can every problem in NP.

2.4.4 NP-Hard

NP-hard (nondeterministic polynomial-time hard), in computational complexity theory, is a class of problems informally "at least as hard as the hardest problems in NP." A problem H is NP-hard if and only if there is an NP-complete problem L that is polynomial time Turing-reducible to H. In other words, L can be solved in polynomial time by an oracle machine with an oracle for H. Informally we can think of an algorithm that can call such an oracle machine as subroutine for solving H, and solves L in polynomial time if the subroutine call takes only one step to compute.

2.4.5 P=NP Question

The question of whether $NP = P$ (can problems that can be solved in non-deterministic polynomial time also always be solved in deterministic polynomial time?) is one of the most important open questions in theoretical computer science and ultra modern mathematics because of the wide implications of a solution. If the answer is yes, many important problems can be shown to have more efficient

solutions that are now used with reluctance because of unknown edge cases. These include various types of integer programming in operations research, many problems in logistics, protein structure prediction in biology, and the ability to find formal proofs of pure mathematics theorems. The $P = NP$ problem is one of the Millennium Prize Problems proposed by the Clay Mathematics Institute the solution of which is a US\$1,000,000 prize for the first person to provide a solution.

2.4.6 NP- Incomplete

Incomplete problems are those in NP that are neither NP-complete nor in P. In other words, incomplete problems can neither be solved in polynomial time nor are they hard problems. It has been shown that if $P = NP$ is found false then there exist NP-incomplete problems.

2.4.7 Co-NP

Co-NP is the set containing the complement problems (i.e. problems with the yes/no answers reversed) of NP problems. It is believed that the two classes are not equal; however it has not yet been proven. It has been shown that if these two complexity classes are not equal, then it follows that no NP-Complete problem can be in co-NP and no co-NP-Complete problem can be in NP.

2.5 Graph and Matching Problems

A graph G is a pair $G = (V, E)$, where V is finite non-empty set of nodes(vertices) and $E \subseteq V \times V$ is a relation set of ordered pairs (u, v) . An edge between two vertices is denoted by $[u, v]$, consists of pairs (u, v) and (v, u) in the set E . A pair $(u, v) \in E$ is called an arc if pair $(v, u) \notin E$. If all pairs in E are arcs, the graph G is called directed graph. Graph G is called an undirected graph if all pairs in E are edges.

Let $G = (V, E)$ be a graph in which vertex set V can be portioned into two disjoint sets, V_1 and V_2 , and each edge in E has one vertex in V_1 and another in V_2 . In such case G is called bipartite graph. Bipartite graph is denoted by $G = (V_1 \cup V_2, E)$. Otherwise the graph is called non-bipartite graph.

A graph $G = (V, E)$ is called a complete graph if $[u, v] \in E$ for all $u, v \in V$ with $u \neq v$. A bipartite graph $G = (V_1 \cup V_2, E)$ is called complete bipartite graph if each $u \in V_1$ is joined to each $v \in V_2$. A graph $G = (V, E)$ with a function $w: E \rightarrow Z$ is called an edge-weighted graph, where Z is usually the set of positive integers.

Given a graph $G = (V, E)$, a matching M in G is a subset of the edge set E with the property that no two edges of M share the same node. A matching M in Graph G is called a maximum matching if no matching in G exists with cardinality more than that of M . The largest possible cardinality of a matching in a graph with $|V|$ nodes is $\lfloor |V|/2 \rfloor$. When the cardinality of a matching M in a graph $G = (V, E)$ is $\lfloor |V|/2 \rfloor$, M is called complete graph or perfect matching.

2.6 Scheduling

Definition of Scheduling and its components are described in different literatures in different ways. According to Pinedo," scheduling concerns the allocation of

limited resources to tasks over time. It is a decision-making process that has a goal the optimization of one or more objective”, Pinedo [24].

In the words of Carlier and Chretienne [4], “Scheduling is to forecast the processing of a work by assigning resources to tasks and fixing their start times. The different components of scheduling problem are the tasks, the potential constraints, the resources and the objective function. The task must be programmed to optimize a specific objective function. Beside this, sometimes it will be more realistic in practice to consider several criteria”, Carlier et al [4]. Furthermore, it is a decision-making problem that plays an important role in most manufacturing and service industries. Scheduling is applied in procurement and production, in transportation and distribution, and in information processing and communication. A scheduling problem typically uses mathematical optimization techniques or heuristic methods to allocate limited resources to the processing of tasks.

In order to determine satisfactory or optimal schedules, it is helpful to formulate the scheduling problem as a mathematical model. Such a model typically describes a number of important characteristics. One characteristic specifies the number of machines or resources as well as their interrelationships with regard to the configuration, for example, machines set up in series, and machines set up in parallel. A second characteristic of a mathematical model concerns the processing requirements and constraints. These include setup costs and setup times, and precedence constraints between various activities. A third characteristic has to do with the objective that has to be optimized, which may be a single objective or a composite of different objectives. For example, the objective may be a combination of maximizing throughput (which is often equivalent to minimizing setup times) and maximizing the number of orders that are shipped on time.

2.6.1 Machine Environment

There can be a single machine, multiple machines, or in some situation, the number of machines may be unknown in advance. The simplest machine environment is the single machine environment, on which each n job J_i , each consisting of single operation, have to spend a processing time equal to their given processing requirements P_i , $i=1, 2, \dots, n$. In case of multiple machine environments, Blazewicz [1], a job J_i , is a set of n_i number of operations, O_i . It is not necessary that an arbitrary operation of an arbitrary job can be processed in an arbitrary machine: this restriction inspires to classify the multiple machine environments into two categories: Parallel machine and Dedicated machine.

In parallel machine model, an arbitrary operation O_i of an arbitrary job J_i can be executed in an arbitrary machine M_j . Simply, any machine can execute any operation of any job.

In dedicated machine model, there is a restriction on operations: operations executable on machines is constrained. To be specific, dedicated machine environment has been classified into three categories, viz., flow shop, open shop and job shop.

2.6.2 Some Application Areas of Scheduling

The application of scheduling is seen in diversified sectors of activity. Some application areas in computer science and engineering are described below.

2.6.2.1 Production Scheduling

Scheduling is an important tool for manufacturing and engineering, where it can have a major impact on the productivity of a process. In manufacturing, the

purpose of scheduling is to minimize the production time and costs, by telling a production facility what to make, when, with which staff, and on which equipment. Production scheduling aims to maximize the efficiency of the operation and reduce costs.

Production scheduling tools greatly outperform older manual scheduling methods. These provide the production scheduler with powerful graphical interfaces which can be used to visually optimize real-time work loads in various stages of production, and pattern recognition allows the software to automatically create scheduling opportunities which might not be apparent without this view into the data. For example, an airline might wish to minimize the number of airport gates required for its aircraft, in order to reduce costs, and scheduling software can allow the planners to see how this can be done, by analyzing time tables, aircraft usage, or the flow of passengers.

2.6.2.2 Operation System Design Scheduling

Scheduling is a key concept in computer multitasking and multiprocessing operating system design, and in real-time operating system design. In modern operating systems, there are typically many more processes running than there are CPUs available to run them. Scheduling refers to the way processes are assigned to run on the available CPUs. This assignment is carried out by software known as a scheduler.

In real-time environments, such as mobile devices for automatic control in industry (for example robotics), the scheduler also must ensure that processes can meet deadlines; this is crucial for keeping the system stable. Scheduled tasks are sent to mobile devices and managed through an administrative back end.

Beside this, some basic algorithms used in OS for uni-processor computers are given below.

i. First Come First Serve (FCFS): At any instance when machine is idle, select the available jobs in the order they request. When the first job enters in the system it is started immediately and allowed to run as long as it wants.

ii. Shortest Job First (SJF): At any instance when the machine is idle, select the available job having shortest expected processing time. In the case of tie the FCFS is used.

iii. Shortest Remaining Time Next (SRTN): At any instance schedule the job whose remaining time is the shortest. When a new job arrives, its time is compared with the current process' remaining time. If new job needs less time to finish than the current process, the current process is suspended and new job started. It is applicable to preemptive system.

iv. Round-Robin: Each process is assigned a time interval, called quantum, which it is allowed to run. If the process is still running at the end of the quantum, the CPU is preempted and given to another process. If the process has finished before the quantum has elapsed, the CPU switching is done when the process blocks, of course.

2.6.2.3 I/O Scheduling

I/O scheduling is the term used to describe the method computer operating systems decide the order that block I/O operations will be submitted to the disk subsystem. I/O scheduling is sometimes called 'disk scheduling'. I/O scheduling usually has to work with hard disks which share the property that there is long access time for requests which are far away from the current position of the disk head (this operation is called a seek). To minimize the effect this has on system performance, most I/O schedulers implement a variant of the elevator algorithm

which re-orders the incoming randomly ordered requests into the order in which they will be found on the disk.

2.6.2.4 Timetable Scheduling

In timetable scheduling problems, examination subjects must be slotted to certain times that satisfy several of constraints. They are NP-completeness problems, which usually lead to satisfactory but suboptimal solutions. Along with this, Timetable scheduling problems concern all educational establishments or universities, since they involve timetabling of courses assuring the availability of teachers, students and classrooms. These problems are just as much the object of studies.

2.6.2.5 Project Scheduling

Project scheduling problems comprise a vast literature. We are interested more generally in problems of scheduling operations which use several resources simultaneously (money, personnel, equipment, raw materials etc.), these resources being available in known amounts. In other words, we deal with the multi-resource scheduling problem with cumulative and non-renewable resources.

2.7 Application of Just-in-Time

The followings are some of the application areas of JIT:

2.7.1 Real Time Operating System

Real Time Operating Systems are dedicated to some well-defined jobs which require very fast response time. This system must be fault-tolerant that is OS must handle the error without going to unstable stage. The execution time is the most

critical issue in real time OS and they must finish the execution of job within pre-defined time-boundary. In a soft-real-time system, early and tardy jobs degrade the quality of the output, while in a hard-real-time system; such jobs make the output invalid. An introduction for real-time scheduling problem in computer system is explained in [27].

2.7.2 Scheduling in Operating System

Scheduling is the key to multiprogramming. Its role is to assign processes to be executed so that some criteria on efficiency are met. Scheduling theory is excessively used in computer manufacturing to schedule the jobs in CPU, memory, printing buffer and other devices for processing jobs. The multiprogramming characteristic of computer is due to the good scheduling of jobs in the CPU because the CPU can only process the job at a time. In this case the objective function is to maximize the CPU utilization (see [28]).

2.7.3 Just-in-Time Compilation

In computing, Just-in-Time, also known as dynamic translation for improving the runtime performance of a computer program. It converts, at runtime, code from one format into another, for example bytecode into native machine code. The performance improvement originates from caching the results of translating blocks of code, not simply evaluating each line or operand separately, or compiling the code at development time. JIT builds upon two earlier ideas in runtime environments: *bytecode compilation and dynamic compilation* (see **Error! Reference source not found.**).

2.7.4 Just-in-Time Sensor Networks

Many areas of research in sensor networks deal directly with the ability to adapt to changing conditions. This has resulted in the ability to dynamically change attributes such as routing paths, MAC protocols, program images, and duty cycling. Yet there are several sensor network optimizations and adaptations that cannot be accomplished through software changes alone. The lack of hardware capabilities or poor geographic layouts of nodes are characteristics that create upper bounds on the ability of software protocols to optimize communication and coverage capabilities. Specifically, a sensor network is deployed (either randomly or placed in a specific location), sits statically for several months collecting data, and adapts itself through various protocols. Yet this often overlooks potential optimizations gained by adding nodes to the network on-demand and within seconds. This introduces a shift in the traditional outdoor, static sensor network paradigm by considering the possibilities and limitations of a rapid, just-in-time deployment (see [28]).

2.7.5 Just-in-Time to Enable Optical Networking for Grids

Many of today's compute- and data-intensive e-science applications are looking to Grid-based technologies to meet their high demands. Until recently, the Grid community focused primarily on maximizing the availability, sharing, and utilization of resources such as CPU power and storage. Now, many in the Grid community are starting to regard the network as another vital Grid resource, to be used to provide large, fast data flows with minimal latency and jitter. MCNC Research and Development Institute and North Carolina State University (NCSU) have developed a Just-In-Time control plane, signaling scheme, and various software and hardware components that are synergistic with these needs. This includes an overview of the Just-In-Time control plane and GridJIT service that has been developed for optical networks and describes several related projects (see [29]).

Chapter 3

JIT PRODUCTION SYSTEM

Just-in-time working is also known as "lean manufacturing"(simply, "Lean", is a production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination). The term comes from quality management theory and the goal is to produce high quality products in the most efficient and economical way. The aim of JIT is to deliver the required production items, at the required quality in the required quantities, at the time they are needed. JIT seeks to achieve zero inventories, zero defects, zero breakdowns, elimination of non-value added activities (e.g., setups and lead times) and delivery of production items on time 100% of the time.

Just-in-time is an inventory strategy implemented to improve the return on investment of a business by reducing in-process inventory and its associated carrying costs. In order to achieve JIT, the process must have signals of what is going on elsewhere within the process. This means, that the process is often driven by a series of signals, which can be Kamban, that tell production processes when to make the next part. Kamban are usually 'tickets' but can be simple visual signals, such as the presence or absence of a part on a shelf. When implemented correctly, JIT can lead to dramatic improvements in a manufacturing organization's return on investment, quality, and efficiency.

Just-in-Time has been implemented in mixed-model assembly line or flexible assembly processes in order to increase profit by reducing cost, and have been used for controlling such flexible assembly system. The intention of these methods is to satisfy the customer demands for a variety of models without holding large inventories or incurring large shortages of the products, Dhamala and Khadka [6]. The most important optimization problem that has to be solved

for the mixed models, just-in-time systems is to determine the sequence in which different models are produced. A great deal of research has been going on on JIT system Monden [23]. The quantity of each part used by the mixed-model assembly line per unit of time should be kept as constant as possible Miltenburg and Sinnamon [21]. Monden [23] states this as the most important goal of a JIT production system implemented by the Toyota Company. Toyota's so-called Goal Chasing Method, a local search heuristic, has been most popular for solving the problem. The sequences referred to as level, balanced or fair sequences always keep the actual production level and the desired production level as close to each other as possible all the times.

The philosophy of JIT is simple - inventory is defined to be waste. JIT inventory systems expose the hidden causes of inventory keeping and are therefore not a simple solution a company can adopt; there is a whole new way of working the company must follow in order to manage its consequences. The ideas in this way of working come from many different disciplines including statistics, industrial engineering, production management and behavioral science. It is more popularized now a days because of its computer applications like real time system and networking. In the JIT inventory philosophy there are views with respect to how inventory is looked upon, what it says about the management within the company, and the main principle behind JIT. Inventory is seen as incurring costs, or waste, instead of adding value, contrary to traditional accounting. This does not mean to say JIT is implemented without awareness that removing inventory exposes pre-existing manufacturing issues. Under this way of working, businesses are encouraged to eliminate inventory that does not compensate for manufacturing issues, and then to constantly improve processes so that less inventory can be kept. Secondly, allowing any stock habituates the management to stock keeping and it can then be a bit like a narcotic. Management is then tempted to keep stock there to hide problems within the production system. These problems include backups at work centers, machine reliability, process variability, lack of flexibility of employees and equipment, and inadequate capacity among other things. In

short, the just-in-time inventory system is all about having “the right material, at the right time, at the right place, and in the exact amount”, without the safety net of inventory. The JIT system has implications of which are broad for the implementers.

3.1 Kamban-an Integrated JIT System

Most Japanese manufacturing companies view the making of a product as continuous from design, manufacture, and distribution to sales and customer service. For many Japanese companies the heart of this process is the Kamban, a Japanese term for "visual record", which directly or indirectly drives much of the manufacturing organization. It was originally developed at Toyota in the 1950s as a way of managing material flow on the assembly line. Over the past three decades the Kamban process, identified as "a highly efficient and effective factory production system", has developed into an optimum manufacturing environment leading to global competitiveness.

The Japanese Kamban process of production is sometimes incorrectly described as a simple just-in-time management technique, a concept which attempts to maintain minimum inventory. The Japanese Kamban process involves more than fine tuning production and supplier scheduling systems, where inventories are minimized by supplying these when needed in production and work in progress is closely monitored. It also encourages; Industrial re-engineering, such as a 'module and cellular production' system, and, Japanese human resources management, where team members are responsible for specific work elements and employees are encouraged to effectively participate in continuously improving Kamban processes within the Kaizen concept.

3.2 Kamban—a Communication Tool in JIT Production System

Kamban has become synonymous with the JIT production system because it has become a very important tool for just-in-time production. Kamban, meaning label or signboard, is used as a communication tool in JIT system. A Kamban is attached to each box of parts as they go to the assembly line. A worker from the following process goes to collect parts from the previous process leaving a kamban signifying the delivery of a given quantity of specific parts. Having all the parts funneled to the line and used as required, the same kamban is returned back to serve as both a record of work done and an order for new parts. Thus Kamban coordinates the inflow of parts and components to the assembly line, minimizing the processes.

3.3 Push versus Pull production system

- **Push System:** total demand is forecast, and the producer allocates (“pushes”) items to user based on the expected needs of all users. Finished goods accumulate in inventory. It is known as “Produce for Forecast”.
- **Pull System:** each user requests (“pulls”) items from the producer only as they are required. Units are only produced if there is demand for them. It is known as “Produce for Demand”.

Current pull systems - JIT, Quick Response, Efficient Consumer Response, and Continuous Replacement.

3.4 Objective of Just-in-Time

Just-in-Time is the name used to describe a manufacturing system where the parts which are needed to complete the finished products are produced or arrive at the assembly site as they are needed.

- Increasing the organization's ability to compete with others and remain competitive over the long run. The competitiveness of the firms is increased by the use of JIT manufacturing process as they can develop a more optimal process for their firms.
- Increasing efficiency within the production process. Efficiency is obtained through the increase of productivity and decrease of cost.
- Reducing wasted materials, time and effort. Wastes that do not add value to the products itself should be eliminated. JIT helps significantly in reducing wastes.
- Identify and response to consumers needs. Customers' needs and wants seem to be the major focus for business now, this objective will help the firm on what is demanded from customers, and what is required of production.
- Optimal quality/cost relationship. The organization should focus on zero-defect production process. Although it seems to be unrealistic, in the long run, it will eliminate a huge amount of resources and effort in inspecting, reworking and the production of defected goods.
- Develop a reliable relationship between the suppliers. A good and long-term relationship between organization and its suppliers helps to manage a more efficient process in inventory management, material management and delivery system. It will also assure that the supply is stable and available when needed.

- Adopt the work ethic of Japanese workers for continuous improvement. Commit a long-term continuous improvement throughout the organization. It will help the organization to remain competitive in the long run.

- Plant design for maximizing efficiency. The design of plant is essential in terms of manufacturing efficiency and utility of resources.

3.5 Toyota Production System

In post-World War II Japan, the founder of Toyota, Sakichi Toyoda, his son Kiichiro Toyoda, and their chief engineer, Taiichi Ohno, developed the Toyota Production System (TPS). TPS is the philosophy that still organizes manufacturing and logistics at Toyota, including the interaction with suppliers and customers. The Toyota Production System refers to an integrated socio-technical system that comprises its management philosophy and practices. The TPS organizes manufacturing and logistics for the automobile manufacturer, including interaction with suppliers and customers. The system is a major precursor of the more generic "Lean manufacturing". The main objectives of the TPS are to design out overburden and inconsistency, and to eliminate waste.

3.6 Lean Manufacturing

Lean Manufacturing, also called Lean Production, is a set of tools and methodologies that aims for the continuous elimination of all waste in the production process. Lean is a business system and philosophy approach to identifying and eliminating waste (non-value-added activities) through continuous process improvement by following the product at the pull of the customer. The goal of Lean is to turn continuous process improvement into a competitive weapon. Lean is all about shortening order to delivery times, lowering costs,

adding higher quality and becoming more flexible simultaneously. Lean can have immediate positive impact on a company. Lean offers many advantages in material handling, inventory, quality, scheduling, personnel and customer satisfaction.

Following are the Objectives of Lean Manufacturing.

- **Defects and wastage** - Reduce defects and unnecessary physical wastage, including excess use of raw material inputs, preventable defects, costs associated with reprocessing defective items, and unnecessary product characteristics which are not required by customers.

- **Cycle times** - Reduce manufacturing lead times and production cycle times by reducing waiting times between processing stages, as well as process preparation times and product/model conversion times.

- **Inventory levels** - Minimize inventory levels at all stages of production, particularly works-in-progress between production stages. Lower inventories also mean lower working capital requirements.

- **Labor productivity** - Improve labor productivity, both by reducing the idle time of workers and ensuring that when workers are working, they are using their effort as productively as possible (including not doing unnecessary tasks or unnecessary motions).

- **Utilization of equipment and space** - Use equipment and manufacturing space more efficiently by eliminating bottlenecks and maximizing the rate of production through existing equipment, while minimizing machine downtime.

- **Flexibility** - Have the ability to produce a more flexible range of products with minimum changeover costs and changeover time.

- **Output** – Insofar as reduced cycle times, increased labor productivity and elimination of bottlenecks and machine downtime can be achieved, companies can generally significantly increased output from their existing facilities.

3.7 Mixed-Model Production System

The increasing market demand for product variety forces manufacturers to design mixed- model assembly lines on which different product models can be switched back and forth and mixed together with little changeover costs. Furthermore, Mixed-model production is the practice of assembling several distinct models of a product on the same assembly line with little changeover costs and then sequencing those models in a way that smoothes the demands for upstream components.

Mixed-Model JIT assembly systems are a fundamental part of the well known “Toyota Production System”. Mixed-Model assembly lines are used to produce many different products without carrying large inventories or incurring large shortages. The effective utilization of these lines requires that a schedule for assembling the different products be defined. Each product assembled on the mixed model assembly line requires variety of parts. Often these parts vary from product to product. Scheduling large lots of each product requires large lots of parts. When a part is only needed for certain products, its usage will be high when those products are being assembled and will be low otherwise. This is that Just-in-Time systems wish to avoid. Just-in-Time systems only work when there is constant rate of usage of all parts. To minimize the variation of usage in each part, products will be sequenced in very small number and mix of parts. In this case we can achieve constant rate of part usage by considering only the demand rates for the products. The objective is then to schedule a constant rate of production for each product.

3.8 Mathematical Model Formulation

When production system consists of constant rate of usage of all parts, Just-in-time systems are suitable. However, the variability between the actual and the ideal production due to integral nature of production appears. This leads the sequencing problem to minimize the variation so that a balanced sequence of diversified products that minimizes the earliness and tardiness penalties could be obtained in a reasonable time. Before starting problem formulation, we assume that the systems have sufficient capacity, negligible switch-over cost and production in unit time. Kubiak [14] refers to single level problem as Product Rate Variation (PRV) problem and multi level problem as Output Rate Variation (ORV) problem.

3.8.1 The PRV Problem Formulation

In Product Rate Variation (PRV) problem, Miltenburg assumes product require approximately the same number and mix of parts. This is a single level case.

Let D units of n products be produced to meet the demands d_i where $i=1, 2, \dots, n$ and $D = \sum_{i=1}^n d_i$ during a specified time horizon. The objective is to maintain cumulative production x_{ik} , a non-negative integer, $i=1, 2, \dots, n$ and $k=1, 2, \dots, D$ of product i during time period 1 through k as close to ideal production kr_i , a non-negative rational number, $i=1, 2, \dots, n$ and $k=1, 2, \dots, D$ with $r_i = d_i/D$ with $\sum_{i=1}^n r_i = 1$ as possible. The specified time horizon is portioned into D equal times of which one unit time is required for a unit of a product to be produced.

The mathematical model of the PRV problem P_1 is as follows:

$$\text{minimize } \left[F = \max_{i,k} f_i(x_{i,k} - kr_i) \right] \quad (3.1)$$

and

$$\text{minimize } \left[G = \sum_{k=1}^D \sum_{i=1}^n f_i(x_{i,k} - kr_i) \right] \quad (3.2)$$

subject to

$$\sum_{i=1}^n x_{i,k} = k, \quad k=1, 2, \dots, D \quad (3.3)$$

$$x_{i,k-1} \leq x_{i,k}, \quad i=1, 2, \dots, n \text{ and } k=1, 2, \dots, D \quad (3.4)$$

$$x_{i,D} = d_i; \quad x_{i,0} = 0, \quad i=1, 2, \dots, n \quad (3.5)$$

$$x_{i,k} \geq 0, \text{ integer} \quad (3.6)$$

The constraint (3.3) shows that exactly k units of products are produced in the periods 1 through k . (3.4) states that the total production is a non-decreasing function of k . (3.5) guarantees the demands are met exactly. (3.3), (3.4) and (3.6) ensure that exactly one unit of a product is sequenced during a time unit.

This model minimizes the perennial objective functions, the bottleneck measure of deviation F that produces smooth sequence in every time unit and the total measure of deviation G (for min-sum) that produces smooth sequence on the average Jost [12].

The exact complexity of the PRV problem still remains open. The problem has been proven to be Co-NP but remains open whether Co-NP-complete or polynomially solvable, Brouner and Crama [2].

3.8.2 The ORV Problem Formulation

The production system consists of hierarchy of several distinct production levels such as products, sub-assemblies, component parts, raw materials, etc. A mixed model multi-level problem falls under ORV problem. Consideration of part demand rate reduces problems into the ORV problem.

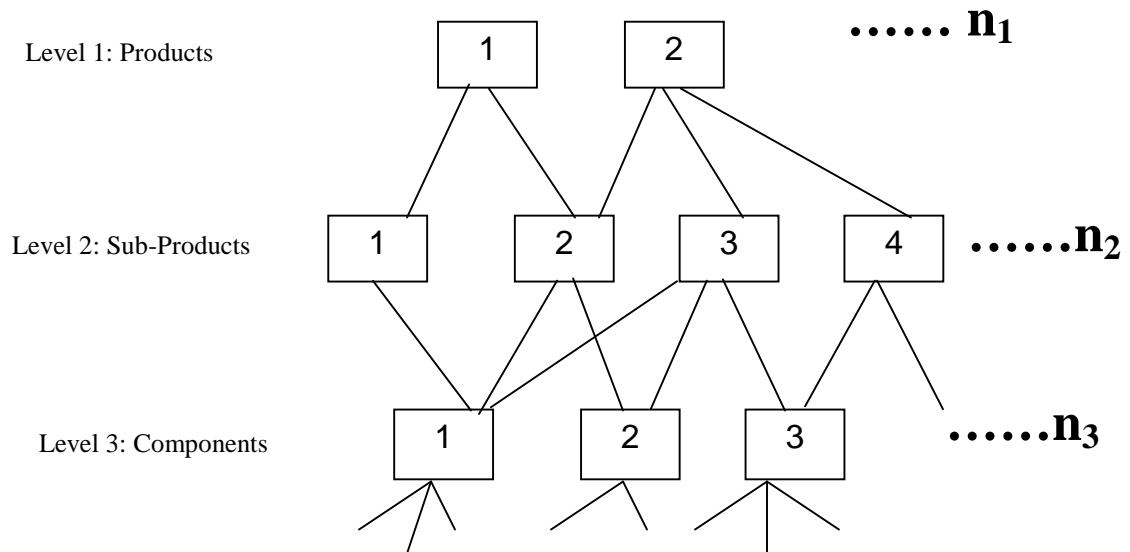


Fig: Mixed-Model Multi-Level Production System

Consider L different production levels $l; l = 1, 2, \dots, L$; where level 1 is the final assembly line. For each $l = 1, 2, \dots, L$; let there be n_l different part types with demands $d_{il}; i = 1, 2, \dots, n_l$. Let t_{ilp} denote the total number of units of output i at

level l required to produce one unit of product p ; $p = 1, 2, \dots, n_1$ so that the dependent demand for part i of level l determined by the final product

demands d_{p1} is $d_{il} = \sum_{p=1}^{n_1} t_{ilp} d_{p1}$. We see that $t_{ilp} = 1$ for $i = 1$ and 0 otherwise. For

each $l = 1, 2, \dots, L$; let $D_l = \sum_{i=1}^{n_l} d_{il}$ be the total output demand of level l . The

demand ratio for part i at level l is $r_{il} = \frac{d_{il}}{D_l}$ for each $i = 1, 2, \dots, n_l$ and we have

$$\sum_{i=1}^{n_l} r_{il} = 1 \text{ for each } i = 1, 2, \dots, n_l.$$

A copy of a product (model) is said to be in stage k ; $k = 1, 2, \dots, D_1$ if k units of products have been produced at level 1. The product level (level 1) has a time horizon of D_1 units and there will be k units of various products p , completely produced, at level 1 during the first k stages. Let the cumulative production of part i at level l during the first k stages be denoted by x_{ilk} so that the total quantity of various parts produced at level l during the first k stages is

$$y_{lk} = \sum_{i=1}^{n_l} x_{ilk} \text{ units. We have } y_{1k} = \sum_{i=1}^{n_1} x_{i1k} = k \text{ at level 1. In fact, } x_{ilk} = \sum_{p=1}^{n_l} t_{ilp} x_{p1k}$$

must hold for $l \geq 2$.

With these notations, the constraints and various objectives for mixed model multi-level JIT assembly systems are formulated as the following [19].

For each $i = 1, 2, \dots, n_l$; let f_{il} be a unimodal, symmetric, convex function with $f_{il}(0) = 0$, minimum. Then the mixed model multi-level JIT scheduling problem defined by (3.7) is to minimize one of the objectives:

$$G_{\max} = \max_{i,l,k} f_{il}(x_{ilk} - y_{lk} r_{il}), \quad (3.7)$$

$$\text{and } G_{\text{sum}} = \sum_{k=1}^{D_1} \sum_{l=1}^L \sum_{i=1}^{n_l} f_{il}(x_{ilk} - y_{lk} r_{il}) \quad (3.8)$$

Subject to the constraints

$$x_{ilk} = \sum_{p=1}^{n_l} t_{ilp} x_{p1k}, \quad i = 1, 2, \dots, n_l; \quad l = 1, 2, \dots, L; \quad k = 1, 2, \dots, D_1 \quad (3.9)$$

$$y_{lk} = \sum_{i=1}^{n_l} x_{ilk}, \quad l = 2, \dots, L; \quad k = 1, 2, \dots, D_1 \quad (3.10)$$

$$y_{1k} = \sum_{i=1}^{n_1} x_{i1k} = k, \quad k = 1, 2, \dots, D_1 \quad (3.11)$$

$$x_{p1k} \geq x_{p1(k-1)}, \quad p = 1, 2, \dots, n_1; \quad k = 1, 2, \dots, D_1 \quad (3.12)$$

$$x_{p1D_1} = d_{p1}, \quad x_{p10} = 0, \quad p = 1, 2, \dots, n_1 \quad (3.13)$$

$$x_{ilk} \geq 0, \text{ integer}, \quad i = 1, 2, \dots, n_l; \quad l = 1, 2, \dots, L; \quad k = 1, 2, \dots, D_1. \quad (3.14)$$

Constraint (3.9) indicates that the necessary cumulative production of part i of level l by the end of stage k is determined explicitly by the quantity of products produced at product level. Constraints (3.10) and (3.11) compute the total cumulative production at level l and level 1, respectively, during the first k stages. Constraint (3.12) shows that the total production of every product over k stages is a non-decreasing function of k . Constraint (3.13) ensures that the production requirements for each product are met exactly. Constraints (3.11), (3.12) and (3.13) indicate that exactly one unit of a product is to be produced in the product level during each stage. ORV problems are NP-hard in general. Two level ORV problems can be solved in pseudo-polynomial time.

Chapter 4

SOLUTION PROCEDURE FOR PRV PROBLEM

The PRV problem is an important production problem that arises on mixed-model assembly lines. The minsum PRV problem consists in sequencing units of different types minimizing the sum of discrepancy functions between the actual and ideal production rates. This problem can be reduced to Assignment Problem (AP) with a matrix of a special structure.

4.1 Release Date/Due Date Decision Problem

To handle large integer programming problems, general solution techniques are not sufficient. A special solution procedure is developed for the specific problem under considerations, Miltenburg [19]. Denote a target value for the objective function by the variable B . The goal is to determine the smallest possible B for which a sequence can be created for each $j(i)$ has a completion time k , such that $f_j^i(k) \leq B$ for $k \in [k_j, (k_{j+1}-1)]$. For target value B , $j(i)$ can not start before $k-1$ if $g_j^i - j - kr_i > B$ and can start k if $f_j^i(k+1) = j - (k+1)r_i \leq B$. Therefore, any fixed target value B allows the calculation of a release date and a due date for a specific copy of a product. For a given B early and late starting dates can be calculated for each copy of each product in a one pass procedure and, hence, can be constructed in $O(D)$ time.

The earliest starting time $E(i, j)$ for (i, j) must be the unique integer satisfying

$$\frac{j-B}{r_i} - 1 \leq E(i, j) < \frac{j-B}{r_i}$$

and latest starting time $L(i, j)$ of (i, j) must be the unique integer satisfying

$$\frac{j-1+B}{r_i} - 1 < L(i, j) \leq \frac{j-1+B}{r_i}$$

formulae:

$$E(i, j) = \left\lceil \frac{j - B}{r_i} - 1 \right\rceil \quad (4.1)$$

and

$$L(i, j) = \left\lfloor \frac{j - 1 + B}{r_i} \right\rfloor \quad (4.2)$$

For a given B , we can determine $E(i, j)$ and $L(i, j)$ for all i and for all j in $O(D)$ time.

4.2 Earliest Due Date Algorithms

In this section we describe a graph theoretic approach for solving the max-abs problem, Steiner and Yeomans [25]. In this procedure, decision version of the problem with certain target value for objective as a threshold value, is reduced to a perfect matching problem in a bipartite graph. Then Glover's modified EDD rule is used for the matching problem to decide whether the decision problem has 'yes' answer. Then an optimal solution is obtained by using the matching problem and bisection search within the bounds for target value after determination of the bounds, Steiner and Yeomans [25].

4.2.1 Perfect Matching Problem and EDD Rule

For a given target value B as threshold value for decision problem, determine $E(i, j)$ and $L(i, j)$ for all i and for all j according to (4.1) and (4.2). Define the bipartite graph $G = (V_1 \cup V_2, E)$;

Where, $V_1 = \{0,1,2,\dots,D-1\}$, $V_2 = \{(i, j) \mid i = 1,2,\dots,n; j = 1,2,\dots,d_i\}$ and $(k, (i, j)) \in E$ if and only if $k \in [E(i, j), L(i, j)]$ i.e. if and only if (i, j) may start at time k . Then the bipartite graph G is V_1 -convex. Here finding a feasible sequence for problem (4.1) is analogous to finding a perfect matching in G such that lower numbered copies of a product are matched to earlier starting times than higher numbered copies. Such a matching is called Order Preserving.

4.2.2 EDD for min-sum-sqr

Inman and Bulfin [11] define the ideal position for copy (i, j) as

$$k_{i,j} = \frac{2j-1}{2r_i} = \left[\frac{\left(j - \frac{1}{2}\right)D}{d_i} \right]$$

Let $Z_{i,j}$ denotes the time at which copy (i, j) actually produced. Then, Inman and Bulfin [11] consider the following problem:

$$\text{Minimize } \sum_{i=1}^n \sum_{j=1}^{d_i} (Z_{i,j} - k_{i,j})^2 \quad (4.3)$$

Subject to

$$Z_{i,j} \leq Z_{i,(j+1)}, \quad i = 1,\dots,n; \quad j = 1,\dots,d_i - 1 \quad (4.4)$$

$$1 \leq Z_{i,j} \leq D, \quad i = 1,\dots,n; \quad j = 1,\dots,d_i \quad (4.5)$$

$$Z_{i,j} \neq Z_{i',j'}, \quad (i, j) \neq (i', j') \quad (4.6)$$

$$Z_{i,j} \in \mathbb{W}, \quad i = 1,\dots,d_i \quad (4.7)$$

Constraint (4.4) ensures that the production time of each copy of a product type i is a strictly increasing function of each copy j . Constraint (4.5) guarantees that the production time of any copy of any product lies in the interval $[1...D]$.

Constraint (4.6) is the only linking constraint and is not in the standard integer programming format and it specifies that only one copy of any product type can be produced in each period. By defining k_{ij} as the due-date of copy (i, j) where each copy of product is treated as a separate job, Inman and Bulfin [11] observe that problem defined by (4.3) may be interpreted as a single machine scheduling problem

$$1|p_{(i,j)} = 1| \sum_{(i,j) \in I} (E_{(i,j)} + T_{(i,j)}), \quad (4.8)$$

where $p_{(i,j)}$, $E_{(i,j)}$ and $T_{(i,j)}$ respectively represents the processing time, earliness and tardiness of copy (i, j) and $I = \{(i, j) | i = 1, \dots, n; j = 1, \dots, d_i\}$.

4.3 Nearest Integer Point Problem

This algorithm aims to minimize the total deviation or sum of all deviation of the real production from the ideal but rational production, Miltenburg [21].

Problem statement

Define the point $X_k = (x_1, x_2, \dots, x_n) \in R_n$ where $x_{i,k} = kr_i$, $\sum_{i=1}^n x_{i,k} = k$, and R

is the set of real number. Problem is to find the “nearest” integer point $M_k =$

$(m_{1,k}, m_{2,k}, \dots, m_{n,k}) \in Z^n$ to the point M_k where $\sum_{i=1}^n m_{i,k} = k$, Z is the

set of nonnegative integers and “nearest” means minimize $\sum_{i=1}^n (m_{i,k} - x_{i,k})^2$

Algorithm 1

The following algorithm finds the nearest integer point $M = (m_{1,k}, m_{2,k}, \dots, m_{n,k}) \in Z^n$ to a point $X = (x_1, x_2, \dots, x_n) \in R_n$ where

$$\sum_{i=1}^n m_i = \sum_{i=1}^n x_i = k.$$

1. Calculate $k = \sum_{i=1}^n x_i$
2. Find the nearest nonnegative integer m_i to each coordinate x_i . That is, find m_i so that $|m_i - x_i| \leq 0.5$, $i = 1, 2, \dots, n$.
3. Calculate $k_m = \sum_{i=1}^n m_i$
 - a. if $k - k_m = 0$ stop. The nearest integer point is $M = (m_1, m_2, \dots, m_n)$
 - b. if $k - k_m > 0$ go to step 5.
 - c. if $k - k_m < 0$ go to step 6.
4. Find the coordinate x_i , with the smallest $m_i - x_i$. Increment the value of this m_i ;
 $m_i \rightarrow m_{i+1}$. Go to step 3.
5. Find the coordinate x_i , with the largest $m_i - x_i$. Decrease the value of this m_i ;
 $m_i \rightarrow m_{i-1}$

Problem with Algorithm 1

For $X = (30/13, 30/13, 5/13)$ the integer point is $(2, 2, 1)$. Then for $X = (36/13, 36/13, 6/13)$ the integer point is $(3, 3, 0)$. Production schedule is 1, 2, -3 which is impossible as production cannot be destroyed. Hence the schedule is not feasible.

Conclusion

Algorithm-1 may lead to infeasible solution.

Algorithm 2

1. Solve the problem P1 (using Algorithm 1), and determine whether the schedule is feasible. (It is feasible if $m_{i,k} - m_{i,k-1} \geq 0$ for all i, k .) If the schedule is feasible, stop. Otherwise, go to step 2.
2. For the infeasible schedule determined in step 1, find the first (or next) stage l where $m_{i,l} - m_{i,l-1} < 0$. Set $n_i =$ number of product i , for which $m_{i,j} - m_{i,l-1} < 0$. Reschedule stages $l - n_i, l - n_i + 1, \dots, l+1$ by considering all possible sequences that begin with the schedule for stage $l - n_i - 1$ and end with the schedule for stage $l + 1$.
3. Repeat step 2 for other stages where $m_{i,k} - m_{i,k-1} < 0$. Then stop.

Problem with Algorithm 2

In general there are $n! / (n - n_i - 2)!$ possible sequences, each of length $n_i + 2$, to consider for each infeasibility. While total enumeration works for small problems of this type (products where similar part requirements) it does not work well for larger problems, nor for problems where products have differing part requirements.

Algorithm 3

1. Solve problem P1 (using Algorithm-1), and determine whether the schedule is feasible. (It is feasible if $m_{i,k} - m_{i,k-1} \geq 0$ for all i, k .) If the schedule is feasible, stop.
2. For the infeasible schedule determine in step 1, find the first (or next) stage l where $m_{i,l} - m_{i,l-1} < 0$. set $n_i =$ number of products i , for which $m_{i,l} - m_{i,l-1} < 0$, and beginning at stage $l - n_i$ use Heuristic 1 or Heuristic 2 to schedule stages $l - n_i, l - n_i + 1, \dots, l + W$, where $W \geq 0$. $l + W$ is the first stage where the schedule determined by heuristic matches the schedule determined in step 1.

3. Repeat step 2 for other schedule determined in step 1.

Heuristic 1

For a stage k , schedule the product i with the lowest $X_{i,k} - kr_i$.

Heuristic 2

For each stage k :

1. Set $h=1$
2. Tentatively schedule products h to be produced in stage k . Calculate the variation for stage k and call it $V1_h$
3. Schedule the product I with the lowest $x_{i,k} - (k+1)r_i$,
4. Increment h ; $h \rightarrow h + 1$. If $h > n$ go to step 5, otherwise go to step 2
5. Schedule the product h with the lowest V_h .

4.4 Dynamic Programming Algorithm

In this section we discuss a dynamic programming (DP) algorithm that deals with JIT production schedule in mixed model facility. The procedure has considered the joint problem with the two typical goals.

1. Usage Goal: maintaining a constant rate of usage of all items in the facility.
2. Loading Goal: smoothing the work load on the final assembly process to reduce the chance of production delays and stoppages.

In this dissertation, we mainly focus on goal 1 which is more important than goal 2, classical goal.

Let there are n products to be produced with demands d_1, d_2, \dots, d_n in a certain time horizon. The time to produce one unit of product i be denoted by t_i ;

$$i = 1, 2, \dots, n \text{ and put } D = \sum_{i=1}^n d_i, r_i = \frac{d_i}{D}.$$

The specified time horizon be inferred into D time units and during each time period k ; $k = 1, 2, \dots, D$; exactly one unit of a product should be produced. Let $x_{i,k}$ denote the total production of product i over the first k periods; where $0 \leq x_{i,k} \leq d_i$ for all $k = 1, 2, \dots, D$. Then $\sum_{i=1}^n x_{i,k} = k$; $k = 1, 2, \dots, D$ and $x_{i,k}$ is non negative integer for all $i = 1, 2, \dots, n$; $k = 1, 2, \dots, D$.

Suppose that the schedule for the first k stages be determined i.e. $x_{i,k}$ for $i = 1, 2, \dots, n$ be known. Then the usage variability at stage k is

$$U_k = \sum_{i=1}^n (x_{i,k} - kr_i)^2 \quad \text{and the loading variability at stage } k$$

$$\text{is } L_k = \sum_{i=1}^n t_i^2 (x_{i,k} - kr_i)^2.$$

Therefore the problem defined by (4.2) can be formulated as

$$\text{Minimize } \sum_{k=1}^D (\Gamma_U U_k + \Gamma_L L_k)$$

Subjected to the Constraints (3.3) - (3.6)

Where Γ_U , Γ_L are relative weights for the Usage Goal and Loading Goal respectively? So the problem defined by (4.2) is a joint problem.

Let f_k denote the joint variability at stage k . Then

$$\begin{aligned} f_k &= \Gamma_U \sum_{i=1}^n (x_{i,k} - kr_i)^2 + \Gamma_L \sum_{i=1}^n t_i^2 (x_{i,k} - kr_i)^2 \\ &= \sum_{i=1}^n (\Gamma_U + \Gamma_L t_i^2) (x_{i,k} - kr_i)^2 \end{aligned}$$

$$= \sum_{i=1}^n T_i^2 (x_{i,k} - kr_i)^2 ; \text{ Where } T_i^2 = r_U + r_L t_i^2 .$$

Therefore the objective function of the problem defined by (5.2) takes the form:

$$\text{Minimize } \sum_{k=1}^D \sum_{i=1}^n T_i^2 (x_{i,k} - kr_i)^2 ; \text{ where call } T_i, \text{ the implied production time for}$$

period i . Now we consider the DP procedure presented by Miltenburg et al. [22].

Let $d = (d_1, d_2, \dots, d_n)$ be the product requirements vector. Define subsets in a schedule as $X = (x_1, x_2, \dots, x_n)$; where x_i is a non negative integer representing the

production of exactly x_i units of product i , $x_i \leq d_i$ for all i . Let e_i be the i^{th} unit vector; with n entries, having i^{th} entry 1 and remaining all zero. A subset X can

be scheduled in the first k stages if $k \geq |X| = \sum_{i=1}^n x_i$.

Let $f(X)$ be the minimal total variation of any schedule where the products in X

are scheduled (produced) during the first k stages. Let $g(X) = \sum_{j=1}^n T_j^2 (x_j - kr_j)^2$.

The following (DP) recursion (R1) holds for $f(X)$:

$$f(X) = f(x_1, x_2, \dots, x_n) = \min \{ f(X - e_i) + g(X) \mid i = 1, \dots, n; x_i - 1 \geq 0 \}$$

$$f(X) = f(X \mid x_i = 0; i = 1, \dots, n) = f(0, 0, \dots, 0) = 0.$$

Clearly $f(X) \geq 0$ and $g(X \mid x_i = 0; i = 1, \dots, n) = 0$. The following theorem tells about the computational efficiency of the above procedure, Miltenburg et al. [22].

4.5 Cost Assignment Problem

Let Z_{ij} denotes the period in which the copy (i, j) is produced. Then the problem defined by (3.2) can be restated as

$$\text{minimize } F_{sum} = \sum_{i=1}^n \left[\sum_{k=0}^{Z_{i1}-1} f_i(0 - kr_i) + \sum_{k=Z_{i1}}^{Z_{i2}-1} f_i(1 - kr_i) + \dots + \sum_{k=Z_{id_i}}^D f_i(d_i - kr_i) \right] \quad (4.9)$$

such that

$$Z_{i,j+1} \geq Z_{ij} + 1, \quad j = 1, \dots, d_i; \quad i = 1, \dots, n \quad (4.10)$$

$$1 \leq Z_{ij} \leq D, \quad j = 1, \dots, d_i; \quad i = 1, \dots, n \quad (4.11)$$

$$Z_{ij} \neq Z_{i'j'}, \quad \text{for } (i, j) \neq (i', j'), \quad Z_{ij} \geq 0 \quad (4.12)$$

Note that constraint (4.12) is the only linking constraint in problem defined by (4.9), which aims specify that only copy of each product can be produced in each period.

The min-sum PRVP can be reduced to an assignment problem and hence can be solved by Hungarian method. For the corresponding assignment problem, we consider the vertex sets $V_1 = \{(i, j) : i = 1, \dots, n, j = 1, \dots, d_i\}$ and $V_2 = \{1, \dots, D\}$. We now have to calculate the appropriate costs to specify its objective function. More specifically, these costs must be such that the assignment problem with these costs has an optimal solution, which is both optimal and feasible for problem (4.9).

Let $C_{i,j,k}$ denotes the cost of assigning (i, j) to the k^{th} period and let

$$x_{i,j,k} = \begin{cases} 1, & \text{if } (i, j) \text{ is assigned to } k \\ 0, & \text{otherwise} \end{cases}$$

Then the assignment problem is

$$\text{minimize } C = \sum_{i=1}^n \sum_{j=1}^{d_i} \sum_{k=1}^D C_{i,j,k} x_{i,j,k} \quad (4.13)$$

$$\text{such that } \sum_{i=1}^n \sum_{j=1}^{d_i} x_{i,j,k} = 1, \quad k = 1, \dots, D \quad (4.14)$$

$$\sum_{k=1}^D x_{i,j,k} = 1, \quad i = 1, \dots, n, \quad j = 1, \dots, d_i \quad (4.15)$$

Constraints on the assignment problem require that

- a) For each (i, j) in V_1 there is exactly one k in V_2 , i.e. each copy is produced exactly once.
- b) For each k in V_2 , there is exactly one (i, j) in V_1 , i.e. exactly one copy is produced at a time.

But Constraints (4.10) on problem defined by (4.9) requires an additional property that

- c) For any two copies (i, j) and (i, j') of a product i , with $j < j'$, if (i, j) is produced at k and (i, j') is produced at k' then $k < k'$.

4.6 Min-sum-absolute-chain Algorithm with EDD

Given:

1. $\text{chain}_1, \text{chain}_2, \dots, \text{chain}_t, \dots, \text{chain}_m$ with chain constraint defined in section (4.6.1)
2. Calculate d_i^t for $i = 1, 2, \dots, n_t$

$$t = 1, 2, \dots, m$$

3. Introduce a new pseudo-job representing each chain by one job as

$$j_i' = \text{pseudo-job for chain}_i$$

$$d'_i = \text{demand for pseudo job } j'_i$$

$$= \text{length of chain}_i$$

4. Calculate due date value for each pseudo-job j'_i ; $i = 1, 2, \dots, n$ by Steiner and Yeoman[25].
5. Schedule this pseudo-job j'_i using EDD Algorithm of Horn [10].
6. Replace each pseudo-job by the real job of the respective chain such that order is preserved.

4.6.1 Chain constraints

1. Chains are non-overlapping.
2. Cyclic chains are not considered.
3. Chains are considered to be optimal sequence.

Example

Input:

Chain1: ababab

Chain 2: ccdcc

Step 1:

| Pseudo Job | Demand |
|------------|--------|
| J1 | 6 |
| J2 | 5 |

Step 2:

EDD Schedule:

J1-J2-J1-J2-J1-J2-J1-J2-J1-J2-J1

Step 3:

a-c-b-c-a-d-b-c-a-c-b

Chapter 5

IMPLEMENTING AND TESTING

The Proposed min-sum-absolute-chain algorithms mentioned in Chapter 5 has been implemented. The program scripts are written in Java Version 1.6.0. The source codes for these programs are included in Appendix. The input data set (chains) represent the demands of products in the mixed model assembly line manufacturing system.

The lists of all possible sequences for given input chains are generated. Cost for each arrangement is calculated (see Appendix for formula). Finally the most efficient chain is selected based on proposed EDD algorithm.

A. 1. Input Chain

| | |
|-----------|-------|
| Chain 1 = | "ab" |
| Chain 2 = | "cdc" |

Table 1: Input Data

2. Table which includes the steps how given chains are combined.

| | | | | |
|----------|----|-----|------|--------------|
| a | ab | abc | abcd | abcde |
| | ac | acb | acbd | acbde |
| | | acd | acdb | acdbe |
| | | | acdc | acdeb |
| c | cd | cdc | cdea | cdcab |
| | | cda | cdab | cdabc |
| | | | cdac | cdacb |
| | ca | cab | cabd | cabde |
| | | cad | cadc | cadcb |
| | | | cadb | cadbc |

Table 2: Combination of Chains

3. Replacement of Chain with Pseudo-Jobs

| Chain | Pseudo-Job |
|-------|------------|
| ab | Job 1 |
| cdc | Job 2 |

Table 3: Replacement of Chain with Pseudo-Jobs

4. Pseudo-job passed to EDD by Proposed-EDD

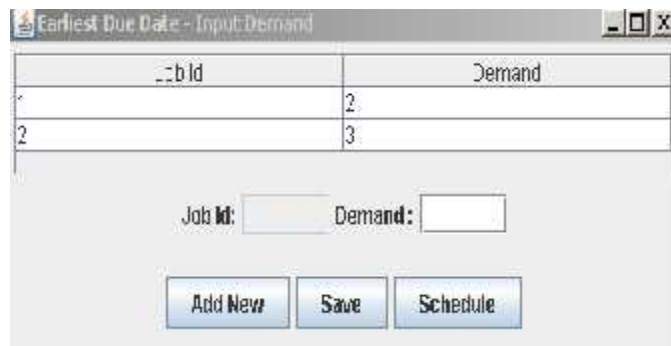


Table 4: Pseudo-job passed to EDD

5. Schedule Generated by EDD with considering the due date $\left[\frac{\left(j - \frac{1}{2} \right) D}{d_i} \right]$ for the j^{th} copy of product i over pseudo-Jobs

| Product | Unit | Due Date |
|---------|------|----------|
| 1 | 1 | 1.25 |
| 1 | 2 | 3.75 |
| 2 | 1 | 0.833 |
| 2 | 2 | 2.5 |
| 2 | 3 | 4.167 |

Schedule List:
2-1-2-1-2-

Table 5: Generation of Schedule by EDD

6. Pseudo-Job generated by EDD Replaced by Proposed-EDD

Pseudo EDD Schedule: cadbc

7. Possible Schedule List generated by Possible Scheduling Algorithm

Sequence No: 1 Evaluating Sequence: abcdc

| | | | | | | |
|----------|------|--------|--------|------------|-------------------|-------------------|
| Index :1 | J :a | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value 2.25 |
| Index :2 | J :b | pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:2.5 |
| Index :3 | J :c | pos :1 | R :0.4 | Zval :1.25 | Cur Value :3.0625 | CumValue:5.5625 |
| Index :4 | J :d | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :7.8125 |
| Index :5 | J :c | pos :2 | R :0.4 | Zval :3.75 | Cur Value :1.5625 | Cum Value :9.375 |

Total Cost: 9.375

Sequence No: 2 Evaluating Sequence: acbdc

| | | | | | | |
|----------|------|--------|--------|------------|-------------------|-------------------|
| Index :1 | J :a | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :2.25 |
| Index :2 | J :c | pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.5625 | Cum Value :2.8125 |
| Index :3 | J :b | pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value :3.0625 |
| Index :4 | J :d | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :5.3125 |
| Index :5 | J :c | pos :2 | R :0.4 | Zval :3.75 | Cur Value :1.5625 | Cum Value :6.875 |

Total Cost: 6.875

Sequence No: 3 Evaluating Sequence: acdbc

| | | | | | | |
|----------|------|--------|--------|------------|-------------------|-------------------|
| Index :1 | J :a | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :2.25 |
| Index :2 | J :c | pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.5625 | Cum Value :2.8125 |
| Index :3 | J :d | pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value :3.0625 |
| Index :4 | J :b | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :5.3125 |
| Index :5 | J :c | pos :2 | R :0.4 | Zval :3.75 | Cur Value :1.5625 | Cum Value :6.875 |

Total Cost: 6.875

Sequence No: 4 Evaluating Sequence: acdcb

| | | | | | | |
|----------|------|--------|--------|------------|-------------------|-------------------|
| Index :1 | J :a | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :2.25 |
| Index :2 | J :c | pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.5625 | Cum Value :2.8125 |
| Index :3 | J :d | pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value :3.0625 |
| Index :4 | J :c | pos :2 | R :0.4 | Zval :3.75 | Cur Value :0.0625 | Cum Value :3.125 |
| Index :5 | J :b | pos :1 | R :0.2 | Zval :2.5 | Cur Value :6.25 | Cum Value :9.375 |

Total Cost: 9.375

Sequence No: 5 Evaluating Sequence: cabdc

| | | | | | | |
|----------|------|--------|--------|------------|-------------------|-------------------|
| Index :1 | J :c | pos :1 | R :0.4 | Zval :1.25 | CurValue :0.0625 | Cum Value :0.0625 |
| Index :2 | J :a | pos :1 | R :0.2 | Zval :2.5 | CurValue :0.25 | Cum Value :0.3125 |
| Index :3 | J :b | pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value :0.5625 |
| Index :4 | J :d | pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value :2.8125 |
| Index :5 | J :c | pos :2 | R :0.4 | Zval :3.75 | Cur Value :1.5625 | Cum Value :4.375 |

Total Cost: 4.375

Sequence No: 6 Evaluating Sequence: cadbc

| | | | | | |
|----------|-------------|--------|------------|-------------------|------------------|
| Index :1 | J :c pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.0625 | Cum Value:0.0625 |
| Index :2 | J :a pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.3125 |
| Index :3 | J :d pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.5625 |
| Index :4 | J :b pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value:2.8125 |
| Index :5 | J :c pos :2 | R :0.4 | Zval :3.75 | Cur Value :1.5625 | Cum Value:4.375 |

Total Cost: 4.375

Sequence No: 7 Evaluating Sequence: cadcb

| | | | | | |
|----------|-------------|--------|------------|-------------------|------------------|
| Index :1 | J :c pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.0625 | Cum Value:0.0625 |
| Index :2 | J :a pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.3125 |
| Index :3 | J :d pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.5625 |
| Index :4 | J :c pos :2 | R :0.4 | Zval :3.75 | Cur Value :0.0625 | Cum Value:0.625 |
| Index :5 | J :b pos :1 | R :0.2 | Zval :2.5 | Cur Value :6.25 | Cum Value:6.875 |

Total Cost: 6.875

Sequence No: 8 Evaluating Sequence: cdabc

| | | | | | |
|----------|-------------|--------|------------|-------------------|------------------|
| Index :1 | J :c pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.0625 | Cum Value:0.0625 |
| Index :2 | J :d pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.3125 |
| Index :3 | J :a pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.5625 |
| Index :4 | J :b pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value:2.8125 |
| Index :5 | J :c pos :2 | R :0.4 | Zval :3.75 | Cur Value :1.5625 | Cum Value:4.375 |

Total Cost: 4.375

Sequence No: 9 Evaluating Sequence: cdacb

| | | | | | |
|----------|-------------|--------|------------|-------------------|------------------|
| Index :1 | J :c pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.0625 | Cum Value:0.0625 |
| Index :2 | J :d pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.3125 |
| Index :3 | J :a pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.5625 |
| Index :4 | J :c pos :2 | R :0.4 | Zval :3.75 | Cur Value :0.0625 | Cum Value:0.625 |
| Index :5 | J :b pos :1 | R :0.2 | Zval :2.5 | Cur Value :6.25 | Cum Value:6.875 |

Total Cost: 6.875

Sequence No: 10 Evaluating Sequence: cdcab

| | | | | | |
|----------|-------------|--------|------------|-------------------|------------------|
| Index :1 | J :c pos :1 | R :0.4 | Zval :1.25 | Cur Value :0.0625 | Cum Value:0.0625 |
| Index :2 | J :d pos :1 | R :0.2 | Zval :2.5 | Cur Value :0.25 | Cum Value:0.3125 |
| Index :3 | J :c pos :2 | R :0.4 | Zval :3.75 | Cur Value :0.5625 | Cum Value:0.875 |
| Index :4 | J :a pos :1 | R :0.2 | Zval :2.5 | Cur Value :2.25 | Cum Value:3.125 |
| Index :5 | J :b pos :1 | R :0.2 | Zval :2.5 | Cur Value :6.25 | Cum Value:9.375 |

Total Cost: 9.375

---Min Value---

Minimal Cost-value: 4.375

8. Minimal Possible Schedules

S.No. Possible Sequence Cost-Value

| | | |
|---|------------------|--------------|
| 1 | cabdc --- | 4.375 |
| 2 | cadbc --- | 4.375 |
| 3 | cdabc --- | 4.375 |

Output:

cadbc is the schedule generated by Proposed -EDD is found in Possible sequence list(Feasibility case).Moreover, this sequence is also found in list of minimal-cost sequence(Optimal case).Hence, it is shown empirically that Proposed-EDD is both feasible and optimal.

B.1.Input Chain

| | |
|-----------|---------|
| Chain 1 = | “aba” |
| Chain 2 = | “ccdcc” |

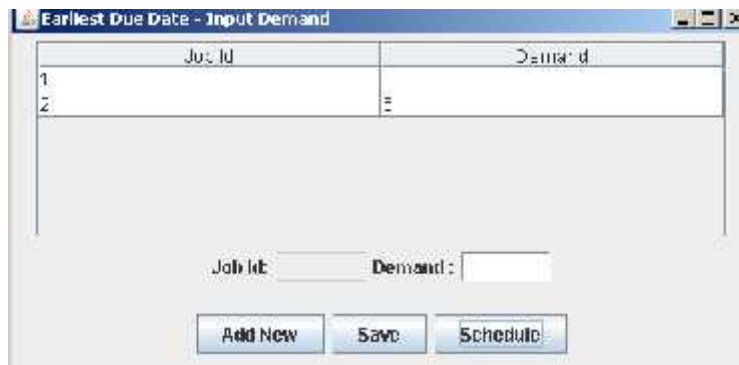
Table 6: Input Data

2. Replacement of Chain with Pseudo-Job

| chain | Pseudo-Job |
|-------|------------|
| aba | Job 1 |
| ccdcc | Job 2 |

Table 7: Replacing Chain with Pseudo-Jobs

3. Pseudo-job passed to EDD by Proposed -EDD



| Job Id | Demand |
|--------|--------|
| 1 | 1 |
| 2 | 1 |

Job Id: Demand:

Table 8: Pseudo-job passed to EDD

4. Schedule Generated by EDD with considering the due date $\left[\frac{\left(j - \frac{1}{2} \right) D}{d_i} \right]$
for the j^{th} copy of product i over pseudo-Jobs

| Product | Unit | Due Date |
|---------|------|----------|
| 1 | 1 | 1.333 |
| 1 | 2 | 4.0 |
| 1 | 3 | 6.667 |
| 2 | 1 | 0.8 |
| 2 | 2 | 2.4 |
| 2 | 3 | 4.0 |
| 2 | 4 | 5.6 |
| 2 | 5 | 7.2 |

Schedule List:
2-1-2-1-2-2-1-2-

Table 9: Generation of Schedule by EDD

5. Pseudo-Job generated by EDD Replaced by Proposed -EDD

Pseudo EDD Schedule: cacbdcac

6. Possible Schedule List generated by Possible Scheduling Algorithm

| S.No. | Possible Sequence | Cost-Value |
|-------|-------------------|------------|
| 1 | abacdcc | 36.0 |
| 2 | abcacdc | 26.0 |
| 3 | abccadcc | 20.0 |
| 4 | abccdacc | 16.0 |
| 5 | abccdca | 14.0 |
| 6 | abccdcca | 16.0 |
| 7 | acbacdcc | 20.0 |
| 8 | acbcadcc | 14.0 |
| 9 | acbcdacc | 10.0 |
| 10 | acbcdca | 8.0 |
| 11 | acbcdcca | 10.0 |
| 12 | acbadcc | 12.0 |
| 13 | accbdacc | 8.0 |
| 14 | accbdca | 6.0 |
| 15 | accbdcca | 8.0 |

| | | |
|----|----------------|------------|
| 16 | acdbacc | 8.0 |
| 17 | acdbca | 6.0 |
| 18 | acdbcca | 8.0 |
| 19 | acdcba | 8.0 |
| 20 | acdcba | 10.0 |
| 21 | acdccba | 16.0 |
| 22 | cabadcc | 18.0 |
| 23 | cabadcc | 12.0 |
| 24 | cabedacc | 8.0 |
| 25 | cabedcac | 6.0 |
| 26 | cabedcca | 8.0 |
| 27 | cabadcc | 10.0 |
| 28 | cabdacc | 6.0 |
| 29 | <i>cacbdca</i> | <i>4.0</i> |
| 30 | cabdcca | 6.0 |
| 31 | caadbacc | 6.0 |
| 32 | <i>caadbca</i> | <i>4.0</i> |
| 33 | caadbcca | 6.0 |
| 34 | caadbca | 6.0 |
| 35 | caadbca | 8.0 |
| 36 | caadbca | 14.0 |
| 37 | ccabadcc | 12.0 |
| 38 | ccabdacc | 8.0 |
| 39 | ccabdcac | 6.0 |
| 40 | ccabdcca | 8.0 |
| 41 | ccadbacc | 8.0 |
| 42 | ccadbca | 6.0 |
| 43 | ccadbcca | 8.0 |
| 44 | ccadbca | 8.0 |
| 45 | ccadbca | 10.0 |
| 46 | ccadbca | 16.0 |
| 47 | ccdabacc | 12.0 |
| 48 | ccdabcac | 10.0 |
| 49 | ccdabcca | 12.0 |
| 50 | ccdabcac | 12.0 |
| 51 | ccdabcba | 14.0 |
| 52 | ccdaccba | 20.0 |
| 53 | ccdcabac | 18.0 |
| 54 | ccdcabca | 20.0 |
| 55 | ccdcacba | 26.0 |
| 56 | ccdccaba | 36.0 |

---Min Value---
Minimal Cost-value: 4.0

7. Minimal Possible Schedules

S.No. Possible Sequence Cost-Value

| | | | |
|---|-----------------|-----|------------|
| 1 | cacbdcac | --- | 4.0 |
| 2 | cacdbcac | --- | 4.0 |

Output:

cacbdcac is the schedule generated by Proposed-EDD is found in Possible sequence list(Feasibility case).Moreover, this sequence is also found in list of minimal-cost sequence(Optimal case). Hence, it is shown empirically that Proposed -EDD is both feasible and optimal.

C.1. Input Chain

| | |
|---------|--------|
| Chain1= | "ab" |
| Chain2= | "ccdc" |
| Chain3= | "mnm" |

Table 10: Input Data

2. Replacement of Chain with Pseudo-Job

| Chain | Pseudo-Job |
|-------|------------|
| Ab | Job1 |
| ccdc | Job2 |
| mnm | Job3 |

Table 11: Replacing Chain with Pseudo-job

3. Pseudo-job passed to EDD by Proposed -EDD

| Job ID | Demand |
|--------|--------|
| 1 | 2 |
| 2 | 4 |
| 3 | 3 |

Table 12: Pseudo-job passed to EDD

4. Schedule Generated by EDD with considering the due date

$$\left[\frac{\left(j - \frac{1}{2} \right) D}{d_i} \right]$$

for the j^{th} copy of product i over pseudo-Jobs

| Product | Unit | Due Date |
|---------|------|----------|
| 1 | 1 | 2.25 |
| | 2 | 4.75 |
| 2 | 1 | 1.125 |
| | 2 | 3.375 |
| | 3 | 5.625 |
| | 4 | 7.875 |
| 3 | 1 | 1.5 |
| ... | 2 | 4.5 |
| ... | 3 | 7.5 |

Schedule List:
2-1-1-2-1-2-1-1-2-

Table 13: Generation of Schedule by EDD

5. Pseudo-Job generated by EDD Replaced by Proposed -EDD

Pseudo EDD Schedule: cmacndbmc

6. Possible Schedule List generated by Possible Scheduling Algorithm

| S.No. | Possible Sequence | Cost-Value |
|-------|-------------------|------------|
| 1 | abccdcmmn | 63.375 |
| 2 | abccdmcnm | 52.875 |
| 3 | abccdmnncm | 46.875 |
| 4 | abccdmnmc | 45.375 |
| 5 | abccmdcnm | 48.375 |
| 6 | abccmdncm | 42.375 |
| 7 | abccmdnmc | 40.875 |
| 8 | abccmndcm | 42.375 |
| 9 | abccmndmc | 40.875 |
| 10 | abccmnmcd | 45.375 |
| 11 | abcmcdcnm | 43.875 |
| 12 | abcmcdncm | 37.875 |
| 13 | abcmcdnmc | 36.375 |
| 14 | abcmcdncm | 37.875 |
| 15 | abcmcdnmc | 36.375 |
| 16 | abcmcnmcd | 40.875 |
| 17 | abcmncdcm | 37.875 |
| 18 | abcmncdmc | 36.375 |
| 19 | abcmncmcd | 40.875 |
| 20 | abcmnmcdc | 45.375 |
| 21 | abmccdcnm | 45.375 |
| 22 | abmccdnncm | 39.375 |
| 23 | abmccdnmc | 37.875 |
| 24 | abmccndcm | 39.375 |
| 25 | abmccndmc | 37.875 |
| 26 | abmccnmcd | 42.375 |
| 27 | abmcncdcm | 39.375 |
| 28 | abmcncdmc | 37.875 |
| 29 | abmcncmcd | 42.375 |
| 30 | abmcnmcdc | 46.875 |
| 31 | abmncdcm | 45.375 |
| 32 | abmncdmc | 43.875 |
| 33 | abmncmcd | 48.375 |
| 34 | abmncmcdc | 52.875 |
| 35 | abmnmccdc | 63.375 |
| 36 | acbedcmnm | 57.375 |
| 37 | acbedmcnm | 46.875 |
| 38 | acbedmncm | 40.875 |
| 39 | acbedmnmc | 39.375 |
| 40 | acbcmcdcnm | 42.375 |
| 41 | acbcmcdncm | 36.375 |
| 42 | acbcmcdnmc | 34.875 |
| 43 | acbcmndcm | 36.375 |
| 44 | acbcmndmc | 34.875 |
| 45 | acbcmnmcd | 39.375 |
| 46 | acbmcdcnm | 37.875 |
| 47 | acbmcdncm | 31.875 |
| 48 | acbmcdnmc | 30.375 |
| 49 | acbmcdncm | 31.875 |

| | | | |
|-----|-----------|-----|--------|
| 50 | acbmcmdmc | --- | 30.375 |
| 51 | acbmcmndc | --- | 34.875 |
| 52 | acbmncdem | --- | 31.875 |
| 53 | acbmncdmc | --- | 30.375 |
| 54 | acbmncmdc | --- | 34.875 |
| 55 | acbmnmcdc | --- | 39.375 |
| 56 | accbdcmm | --- | 57.375 |
| 57 | accbdcnm | --- | 46.875 |
| 58 | accbdmncm | --- | 40.875 |
| 59 | accbdmnm | --- | 39.375 |
| 60 | accbmdcnm | --- | 42.375 |
| 61 | accbmdncm | --- | 36.375 |
| 62 | accbmdnmc | --- | 34.875 |
| 63 | accbmndcm | --- | 36.375 |
| 64 | accbmndmc | --- | 34.875 |
| 65 | accbmnmcd | --- | 39.375 |
| 66 | accdbcmnm | --- | 57.375 |
| 67 | accdbcmnm | --- | 46.875 |
| 68 | accdbmncm | --- | 40.875 |
| 69 | accdbmnm | --- | 39.375 |
| 70 | accdcbmnm | --- | 63.375 |
| 71 | accdcmnbm | --- | 58.875 |
| 72 | accdcmnmb | --- | 58.875 |
| 73 | accdcmnmb | --- | 63.375 |
| 74 | accdmcbnm | --- | 42.375 |
| 75 | accdmcbnm | --- | 36.375 |
| 76 | accdmbnmc | --- | 34.875 |
| 77 | accdmcbnm | --- | 48.375 |
| 78 | accdmcnmb | --- | 48.375 |
| 79 | accdmcnmb | --- | 52.875 |
| 80 | accdmnbcm | --- | 36.375 |
| 81 | accdmnbmc | --- | 34.875 |
| 82 | accdmncbm | --- | 42.375 |
| 83 | accdmncmb | --- | 46.875 |
| 84 | accdmnmcb | --- | 39.375 |
| 85 | accdmnmcb | --- | 45.375 |
| 86 | accmbdcnm | --- | 37.875 |
| 87 | accmbdncm | --- | 31.875 |
| 88 | accmbdncm | --- | 30.375 |
| 89 | accmbndcm | --- | 31.875 |
| 90 | accmbndmc | --- | 30.375 |
| 91 | accmbnmcd | --- | 34.875 |
| 92 | accmdbcnm | --- | 37.875 |
| 93 | accmdbcnm | --- | 31.875 |
| 94 | accmdbnmc | --- | 30.375 |
| 95 | accmdcbnm | --- | 43.875 |
| 96 | accmdcnbm | --- | 43.875 |
| 97 | accmdcnmb | --- | 48.375 |
| 98 | accmdnbcm | --- | 31.875 |
| 99 | accmdnbmc | --- | 30.375 |
| 100 | accmdncbm | --- | 37.875 |
| 101 | accmdncmb | --- | 42.375 |
| 102 | accmdnmcb | --- | 34.875 |
| 103 | accmdnmcb | --- | 40.875 |
| 104 | accmnbdc | --- | 31.875 |
| 105 | accmnbdc | --- | 30.375 |

| | | | |
|-----|-----------|-----|--------|
| 106 | accmnbmdc | --- | 34.875 |
| 107 | accmndbcm | --- | 31.875 |
| 108 | accmndbmc | --- | 30.375 |
| 109 | accmndcbm | --- | 37.875 |
| 110 | accmndcmb | --- | 42.375 |
| 111 | accmndmbc | --- | 34.875 |
| 112 | accmndmcb | --- | 40.875 |
| 113 | accmnmbdc | --- | 39.375 |
| 114 | accmnmdbc | --- | 39.375 |
| 115 | accmnmdcb | --- | 45.375 |
| 116 | acmbcdcnm | --- | 33.375 |
| 117 | acmbcdncm | --- | 27.375 |
| 118 | acmbcdnmc | --- | 25.875 |
| 119 | acmbcndcm | --- | 27.375 |
| 120 | acmbcndmc | --- | 25.875 |
| 121 | acmbcnmdc | --- | 30.375 |
| 122 | acmbncdcm | --- | 27.375 |
| 123 | acmbncdmc | --- | 25.875 |
| 124 | acmbncmdc | --- | 30.375 |
| 125 | acmbnmdc | --- | 34.875 |
| 126 | acmcbdcnm | --- | 33.375 |
| 127 | acmcbdncm | --- | 27.375 |
| 128 | acmcbdnmc | --- | 25.875 |
| 129 | acmcbndcm | --- | 27.375 |
| 130 | acmcbndmc | --- | 25.875 |
| 131 | acmcbnmdc | --- | 30.375 |
| 132 | acmcdbcnm | --- | 33.375 |
| 133 | acmcdbncm | --- | 27.375 |
| 134 | acmcdbnmc | --- | 25.875 |
| 135 | acmcdcbnm | --- | 39.375 |
| 136 | acmcdcnbm | --- | 39.375 |
| 137 | acmcdcnmb | --- | 43.875 |
| 138 | acmcdnbcm | --- | 27.375 |
| 139 | acmcdnbmc | --- | 25.875 |
| 140 | acmcdncbm | --- | 33.375 |
| 141 | acmcdncmb | --- | 37.875 |
| 142 | acmcdnmbc | --- | 30.375 |
| 143 | acmcdnmcb | --- | 36.375 |
| 144 | acmenbdcm | --- | 27.375 |
| 145 | acmenbdmc | --- | 25.875 |
| 146 | acmenbmdc | --- | 30.375 |
| 147 | acmndbcm | --- | 27.375 |
| 148 | acmndbmc | --- | 25.875 |
| 149 | acmndcbm | --- | 33.375 |
| 150 | acmndcmb | --- | 37.875 |
| 151 | acmndmbc | --- | 30.375 |
| 152 | acmndmcb | --- | 36.375 |
| 153 | acmenmbdc | --- | 34.875 |
| 154 | acmenmdbc | --- | 34.875 |
| 155 | acmenmdcb | --- | 40.875 |
| 156 | acmnbedcm | --- | 27.375 |
| 157 | acmnbedmc | --- | 25.875 |
| 158 | acmnbcmdc | --- | 30.375 |
| 159 | acmnbcmdc | --- | 34.875 |
| 160 | acmncbdcm | --- | 27.375 |
| 161 | acmncbdmc | --- | 25.875 |

| | | | |
|-----|------------|-----|--------|
| 162 | acmncbmdc | --- | 30.375 |
| 163 | acmncdbcm | --- | 27.375 |
| 164 | acmncdbmc | --- | 25.875 |
| 165 | acmncdcbm | --- | 33.375 |
| 166 | acmncdcm | --- | 37.875 |
| 167 | acmncdmbc | --- | 30.375 |
| 168 | acmncdmc | --- | 36.375 |
| 169 | acmncmbdc | --- | 34.875 |
| 170 | acmncmdbc | --- | 34.875 |
| 171 | acmncmdcb | --- | 40.875 |
| 172 | acmnmcbdc | --- | 39.375 |
| 173 | acmnmcbdc | --- | 39.375 |
| 174 | acmnmcdbc | --- | 39.375 |
| 175 | acmnmcdcb | --- | 45.375 |
| 176 | ambccdcnm | --- | 40.875 |
| 177 | ambccdcnm | --- | 34.875 |
| 178 | ambccdnmc | --- | 33.375 |
| 179 | ambccndcm | --- | 34.875 |
| 180 | ambccndmc | --- | 33.375 |
| 181 | ambccnmdc | --- | 37.875 |
| 182 | ambcnedcm | --- | 34.875 |
| 183 | ambcnedmc | --- | 33.375 |
| 184 | ambcnemdc | --- | 37.875 |
| 185 | ambcnmedc | --- | 42.375 |
| 186 | ambnccdc | --- | 40.875 |
| 187 | ambnccdm | --- | 39.375 |
| 188 | ambnccmdc | --- | 43.875 |
| 189 | ambnccmde | --- | 48.375 |
| 190 | ambnccede | --- | 58.875 |
| 191 | amcbcdcnm | --- | 34.875 |
| 192 | amcbcdncm | --- | 28.875 |
| 193 | amcbcdnmc | --- | 27.375 |
| 194 | amcbcdndcm | --- | 28.875 |
| 195 | amcbcdndmc | --- | 27.375 |
| 196 | amcbcnmdc | --- | 31.875 |
| 197 | amcbncdcm | --- | 28.875 |
| 198 | amcbncdm | --- | 27.375 |
| 199 | amcbncmde | --- | 31.875 |
| 200 | amcbnmedc | --- | 36.375 |
| 201 | amccbdcnm | --- | 34.875 |
| 202 | amccbdcnm | --- | 28.875 |
| 203 | amccbdcnmc | --- | 27.375 |
| 204 | amccbndcm | --- | 28.875 |
| 205 | amccbndmc | --- | 27.375 |
| 206 | amccbndmc | --- | 31.875 |
| 207 | amccdbcnm | --- | 34.875 |
| 208 | amccdbncm | --- | 28.875 |
| 209 | amccdbnmc | --- | 27.375 |
| 210 | amccdcbnm | --- | 40.875 |
| 211 | amccdcnbnm | --- | 40.875 |
| 212 | amccdcnmb | --- | 45.375 |
| 213 | amccdnbcm | --- | 28.875 |
| 214 | amccdnbcm | --- | 27.375 |
| 215 | amccdnbcm | --- | 34.875 |
| 216 | amccdnmb | --- | 39.375 |
| 217 | amccdnmbc | --- | 31.875 |

| | | | |
|-----|-----------|-----|--------|
| 218 | amccdnmcb | --- | 37.875 |
| 219 | amccnbdc | --- | 28.875 |
| 220 | amccnbdmc | --- | 27.375 |
| 221 | amccnbmdc | --- | 31.875 |
| 222 | amccndbcm | --- | 28.875 |
| 223 | amccndbmc | --- | 27.375 |
| 224 | amccndcbm | --- | 34.875 |
| 225 | amccndcmb | --- | 39.375 |
| 226 | amccndmcb | --- | 31.875 |
| 227 | amccndmcb | --- | 37.875 |
| 228 | amccnmbdc | --- | 36.375 |
| 229 | amccnmdb | --- | 36.375 |
| 230 | amccnmdb | --- | 42.375 |
| 231 | amcnbcdcm | --- | 28.875 |
| 232 | amcnbcdmc | --- | 27.375 |
| 233 | amcnbcmdc | --- | 31.875 |
| 234 | amcnbcmdc | --- | 36.375 |
| 235 | amcnbcdcm | --- | 28.875 |
| 236 | amcnbcdmc | --- | 27.375 |
| 237 | amcnbcmdc | --- | 31.875 |
| 238 | amcnbdbcm | --- | 28.875 |
| 239 | amcnbdbmc | --- | 27.375 |
| 240 | amcnbdbcm | --- | 34.875 |
| 241 | amcnbdbcm | --- | 39.375 |
| 242 | amcnbdbmc | --- | 31.875 |
| 243 | amcnbdbcm | --- | 37.875 |
| 244 | amcnmbdc | --- | 36.375 |
| 245 | amcnmdbc | --- | 36.375 |
| 246 | amcnmdbc | --- | 42.375 |
| 247 | amcnmbedc | --- | 40.875 |
| 248 | amcnmcbdc | --- | 40.875 |
| 249 | amcnmcdbc | --- | 40.875 |
| 250 | amcnmcdcb | --- | 46.875 |
| 251 | amnbccdc | --- | 40.875 |
| 252 | amnbccdm | --- | 39.375 |
| 253 | amnbccm | --- | 43.875 |
| 254 | amnbcmcdc | --- | 48.375 |
| 255 | amnbmccdc | --- | 58.875 |
| 256 | amncbedcm | --- | 34.875 |
| 257 | amncbcdmc | --- | 33.375 |
| 258 | amncbcmdc | --- | 37.875 |
| 259 | amncbcmdc | --- | 42.375 |
| 260 | amncbcdcm | --- | 34.875 |
| 261 | amncbcdmc | --- | 33.375 |
| 262 | amncbcmdc | --- | 37.875 |
| 263 | amncdbcm | --- | 34.875 |
| 264 | amncdbmc | --- | 33.375 |
| 265 | amncdbcm | --- | 40.875 |
| 266 | amncdcmb | --- | 45.375 |
| 267 | amncdmbc | --- | 37.875 |
| 268 | amncdmcb | --- | 43.875 |
| 269 | amncmbdc | --- | 42.375 |
| 270 | amncmdbc | --- | 42.375 |
| 271 | amncmdbc | --- | 48.375 |
| 272 | amncmbedc | --- | 46.875 |
| 273 | amncmcbdc | --- | 46.875 |

| | | | |
|-----|------------|-----|--------|
| 274 | amncmcdbc | --- | 46.875 |
| 275 | amncmcacb | --- | 52.875 |
| 276 | amnmbccdc | --- | 63.375 |
| 277 | amnmbcbdc | --- | 57.375 |
| 278 | amnmccbdc | --- | 57.375 |
| 279 | amnmccdbc | --- | 57.375 |
| 280 | amnmccacb | --- | 63.375 |
| 281 | cabcdcmnm | --- | 51.375 |
| 282 | cabcdmcnm | --- | 40.875 |
| 283 | cabcdmncm | --- | 34.875 |
| 284 | cabcdmnmc | --- | 33.375 |
| 285 | cabcmdecnm | --- | 36.375 |
| 286 | cabcmdncm | --- | 30.375 |
| 287 | cabcmdnmc | --- | 28.875 |
| 288 | cabcmndcm | --- | 30.375 |
| 289 | cabcmndmc | --- | 28.875 |
| 290 | cabcmnmde | --- | 33.375 |
| 291 | cabmcdcnm | --- | 31.875 |
| 292 | cabmcdnmc | --- | 25.875 |
| 293 | cabmcdnmc | --- | 24.375 |
| 294 | cabmcdnmc | --- | 25.875 |
| 295 | cabmcndmc | --- | 24.375 |
| 296 | cabmcnmde | --- | 28.875 |
| 297 | cabmncdcm | --- | 25.875 |
| 298 | cabmncdmc | --- | 24.375 |
| 299 | cabmncmde | --- | 28.875 |
| 300 | cabmnmcdc | --- | 33.375 |
| 301 | cacbdcmnm | --- | 51.375 |
| 302 | cacbdmcnm | --- | 40.875 |
| 303 | cacbdmncm | --- | 34.875 |
| 304 | cacbdmnmc | --- | 33.375 |
| 305 | cacbmdecnm | --- | 36.375 |
| 306 | cacbmdncm | --- | 30.375 |
| 307 | cacbmdnmc | --- | 28.875 |
| 308 | cacbmndcm | --- | 30.375 |
| 309 | cacbmndmc | --- | 28.875 |
| 310 | cacbmnmde | --- | 33.375 |
| 311 | caadbcmnm | --- | 51.375 |
| 312 | caadbmcnm | --- | 40.875 |
| 313 | caadbmncm | --- | 34.875 |
| 314 | caadbmnmc | --- | 33.375 |
| 315 | caadcbmnm | --- | 57.375 |
| 316 | caadcmnbm | --- | 52.875 |
| 317 | caadcmnbm | --- | 52.875 |
| 318 | caadcmnmb | --- | 57.375 |
| 319 | caadmbcnm | --- | 36.375 |
| 320 | caadmbnmc | --- | 30.375 |
| 321 | caadmbnmc | --- | 28.875 |
| 322 | caadmcbnm | --- | 42.375 |
| 323 | caadmcbnm | --- | 42.375 |
| 324 | caadmcbnm | --- | 46.875 |
| 325 | caadmnbcm | --- | 30.375 |
| 326 | caadmnbmc | --- | 28.875 |
| 327 | caadmncbm | --- | 36.375 |
| 328 | caadmncmb | --- | 40.875 |
| 329 | caadmnmcb | --- | 33.375 |

| | | | |
|-----|-----------|-----|--------|
| 330 | cacdmmcb | --- | 39.375 |
| 331 | cacmbdcnm | --- | 31.875 |
| 332 | cacmbdncm | --- | 25.875 |
| 333 | cacmbdnmc | --- | 24.375 |
| 334 | cacmbndcm | --- | 25.875 |
| 335 | cacmbndmc | --- | 24.375 |
| 336 | cacmbnmdc | --- | 28.875 |
| 337 | cacmdbcnm | --- | 31.875 |
| 338 | cacmdbncm | --- | 25.875 |
| 339 | cacmdbnmc | --- | 24.375 |
| 340 | cacmdcbnm | --- | 37.875 |
| 341 | cacmdcnbm | --- | 37.875 |
| 342 | cacmdcnmb | --- | 42.375 |
| 343 | cacmdnbcm | --- | 25.875 |
| 344 | cacmdnbmc | --- | 24.375 |
| 345 | cacmdncbm | --- | 31.875 |
| 346 | cacmdncmb | --- | 36.375 |
| 347 | cacmdnmbc | --- | 28.875 |
| 348 | cacmdnmcb | --- | 34.875 |
| 349 | cacmnbdcn | --- | 25.875 |
| 350 | cacmnbdmc | --- | 24.375 |
| 351 | cacmnbmdc | --- | 28.875 |
| 352 | cacmndbcm | --- | 25.875 |
| 353 | cacmndbmc | --- | 24.375 |
| 354 | cacmndcbm | --- | 31.875 |
| 355 | cacmndcmb | --- | 36.375 |
| 356 | cacmndmbc | --- | 28.875 |
| 357 | cacmndmcb | --- | 34.875 |
| 358 | cacmnmdbc | --- | 33.375 |
| 359 | cacmnmdbm | --- | 33.375 |
| 360 | cacmnmdbm | --- | 39.375 |
| 361 | cambcdcnm | --- | 27.375 |
| 362 | cambcdncm | --- | 21.375 |
| 363 | cambcdnmc | --- | 19.875 |
| 364 | cambcndcm | --- | 21.375 |
| 365 | cambcndmc | --- | 19.875 |
| 366 | cambcnmdc | --- | 24.375 |
| 367 | cambncdcm | --- | 21.375 |
| 368 | cambncdmc | --- | 19.875 |
| 369 | cambncmdc | --- | 24.375 |
| 370 | cambnmdc | --- | 28.875 |
| 371 | camcbdcnm | --- | 27.375 |
| 372 | camcbdncm | --- | 21.375 |
| 373 | camcbdncm | --- | 19.875 |
| 374 | camcbndcm | --- | 21.375 |
| 375 | camcbndmc | --- | 19.875 |
| 376 | camcbnmdc | --- | 24.375 |
| 377 | camcdbenm | --- | 27.375 |
| 378 | camcdbncm | --- | 21.375 |
| 379 | camcdbnmc | --- | 19.875 |
| 380 | camcdcbnm | --- | 33.375 |
| 381 | camcdcnbm | --- | 33.375 |
| 382 | camcdcnmb | --- | 37.875 |
| 383 | camcdnbcm | --- | 21.375 |
| 384 | camcdnbmc | --- | 19.875 |
| 385 | camcdncbm | --- | 27.375 |

| | | | |
|-----|------------|-----|--------|
| 386 | camcdncmb | --- | 31.875 |
| 387 | camcdnmbc | --- | 24.375 |
| 388 | camcdnmcb | --- | 30.375 |
| 389 | camcnbdcm | --- | 21.375 |
| 390 | camcnbdmc | --- | 19.875 |
| 391 | camcnbmdc | --- | 24.375 |
| 392 | camcnbdbcm | --- | 21.375 |
| 393 | camcnbdbmc | --- | 19.875 |
| 394 | camcnbdbcm | --- | 27.375 |
| 395 | camcnbdbmb | --- | 31.875 |
| 396 | camcnbdbmc | --- | 24.375 |
| 397 | camcnbdbcb | --- | 30.375 |
| 398 | camcnmbdc | --- | 28.875 |
| 399 | camcnmdbc | --- | 28.875 |
| 400 | camcnmdbcb | --- | 34.875 |
| 401 | camnbcddcm | --- | 21.375 |
| 402 | camnbcddmc | --- | 19.875 |
| 403 | camnbcddc | --- | 24.375 |
| 404 | camnbcddc | --- | 28.875 |
| 405 | camnbcddcm | --- | 21.375 |
| 406 | camnbcddmc | --- | 19.875 |
| 407 | camnbcddc | --- | 24.375 |
| 408 | camnbcddcm | --- | 21.375 |
| 409 | camnbcddmc | --- | 19.875 |
| 410 | camnbcddcm | --- | 27.375 |
| 411 | camnbcddmb | --- | 31.875 |
| 412 | camnbcddmc | --- | 24.375 |
| 413 | camnbcddcb | --- | 30.375 |
| 414 | camnbcddc | --- | 28.875 |
| 415 | camnbcddc | --- | 28.875 |
| 416 | camnbcddcb | --- | 34.875 |
| 417 | camnbcddc | --- | 33.375 |
| 418 | camnbcddc | --- | 33.375 |
| 419 | camnbcddc | --- | 33.375 |
| 420 | camnbcddcb | --- | 39.375 |
| 421 | ccabdcmm | --- | 51.375 |
| 422 | ccabdcmm | --- | 40.875 |
| 423 | ccabdcmm | --- | 34.875 |
| 424 | ccabdcmm | --- | 33.375 |
| 425 | ccabdcmm | --- | 36.375 |
| 426 | ccabdcmm | --- | 30.375 |
| 427 | ccabdcmm | --- | 28.875 |
| 428 | ccabdcmm | --- | 30.375 |
| 429 | ccabdcmm | --- | 28.875 |
| 430 | ccabdcmm | --- | 33.375 |
| 431 | ccadbcmm | --- | 51.375 |
| 432 | ccadbcmm | --- | 40.875 |
| 433 | ccadbcmm | --- | 34.875 |
| 434 | ccadbcmm | --- | 33.375 |
| 435 | ccadbcmm | --- | 57.375 |
| 436 | ccadbcmm | --- | 52.875 |
| 437 | ccadbcmm | --- | 52.875 |
| 438 | ccadbcmm | --- | 57.375 |
| 439 | ccadbcmm | --- | 36.375 |
| 440 | ccadbcmm | --- | 30.375 |
| 441 | ccadbcmm | --- | 28.875 |

| | | | |
|-----|-----------|-----|--------|
| 442 | ccadmcbnm | --- | 42.375 |
| 443 | ccadmcnbm | --- | 42.375 |
| 444 | ccadmcnmb | --- | 46.875 |
| 445 | ccadmnbcm | --- | 30.375 |
| 446 | ccadmnbmc | --- | 28.875 |
| 447 | ccadmncbm | --- | 36.375 |
| 448 | ccadmncmb | --- | 40.875 |
| 449 | ccadmnmcb | --- | 33.375 |
| 450 | ccadmnmcb | --- | 39.375 |
| 451 | ccambdcnm | --- | 31.875 |
| 452 | ccambdncm | --- | 25.875 |
| 453 | ccambdnmc | --- | 24.375 |
| 454 | ccambndcm | --- | 25.875 |
| 455 | ccambndmc | --- | 24.375 |
| 456 | ccambnmdc | --- | 28.875 |
| 457 | ccamdbcnm | --- | 31.875 |
| 458 | ccamdbncm | --- | 25.875 |
| 459 | ccamdbnmc | --- | 24.375 |
| 460 | ccamdcbnm | --- | 37.875 |
| 461 | ccamdcnbm | --- | 37.875 |
| 462 | ccamdcnmb | --- | 42.375 |
| 463 | ccamdnbcm | --- | 25.875 |
| 464 | ccamdnbmc | --- | 24.375 |
| 465 | ccamdncbm | --- | 31.875 |
| 466 | ccamdncmb | --- | 36.375 |
| 467 | ccamdnmcb | --- | 28.875 |
| 468 | ccamdnmcb | --- | 34.875 |
| 469 | ccamnbdcn | --- | 25.875 |
| 470 | ccamnbdcn | --- | 24.375 |
| 471 | ccamnbmdc | --- | 28.875 |
| 472 | ccamndbcn | --- | 25.875 |
| 473 | ccamndbmc | --- | 24.375 |
| 474 | ccamndcbm | --- | 31.875 |
| 475 | ccamndcmb | --- | 36.375 |
| 476 | ccamndmcb | --- | 28.875 |
| 477 | ccamndmcb | --- | 34.875 |
| 478 | ccamnmdbc | --- | 33.375 |
| 479 | ccamnmdbc | --- | 33.375 |
| 480 | ccamnmdeb | --- | 39.375 |
| 481 | ccdabcnmn | --- | 51.375 |
| 482 | ccdabmcnm | --- | 40.875 |
| 483 | ccdabmncm | --- | 34.875 |
| 484 | ccdabmnmc | --- | 33.375 |
| 485 | ccdacbnm | --- | 57.375 |
| 486 | ccdacmbnm | --- | 52.875 |
| 487 | ccdacmnbm | --- | 52.875 |
| 488 | ccdacmnm | --- | 57.375 |
| 489 | ccdambcnm | --- | 36.375 |
| 490 | ccdambnmc | --- | 30.375 |
| 491 | ccdambnmc | --- | 28.875 |
| 492 | ccdamcbnm | --- | 42.375 |
| 493 | ccdamcnbm | --- | 42.375 |
| 494 | ccdamcnmb | --- | 46.875 |
| 495 | ccdamnbcn | --- | 30.375 |
| 496 | ccdambnmc | --- | 28.875 |
| 497 | ccdambnmc | --- | 36.375 |

| | | | |
|-----|------------|-----|--------|
| 498 | ccdammcmb | --- | 40.875 |
| 499 | ccdammnmbc | --- | 33.375 |
| 500 | ccdammnmb | --- | 39.375 |
| 501 | ccdcabnmnm | --- | 63.375 |
| 502 | ccdcambnm | --- | 58.875 |
| 503 | ccdcamnbnm | --- | 58.875 |
| 504 | ccdcamnmb | --- | 63.375 |
| 505 | ccdcmanbnm | --- | 54.375 |
| 506 | ccdcmanbnm | --- | 54.375 |
| 507 | ccdcmanmb | --- | 58.875 |
| 508 | ccdcmanabm | --- | 54.375 |
| 509 | ccdcmanamb | --- | 58.875 |
| 510 | ccdcmanmab | --- | 63.375 |
| 511 | ccdmabcnm | --- | 31.875 |
| 512 | ccdmabncm | --- | 25.875 |
| 513 | ccdmabnmc | --- | 24.375 |
| 514 | ccdmacbnm | --- | 37.875 |
| 515 | ccdmacnbnm | --- | 37.875 |
| 516 | ccdmacnmb | --- | 42.375 |
| 517 | ccdmancbnm | --- | 25.875 |
| 518 | ccdmancbmc | --- | 24.375 |
| 519 | ccdmancbnm | --- | 31.875 |
| 520 | ccdmancmb | --- | 36.375 |
| 521 | ccdmancmb | --- | 28.875 |
| 522 | ccdmancmb | --- | 34.875 |
| 523 | ccdmcanbnm | --- | 43.875 |
| 524 | ccdmcanbnm | --- | 43.875 |
| 525 | ccdmcanmb | --- | 48.375 |
| 526 | ccdmcanabm | --- | 43.875 |
| 527 | ccdmcanamb | --- | 48.375 |
| 528 | ccdmcanmab | --- | 52.875 |
| 529 | ccdmnabcnm | --- | 25.875 |
| 530 | ccdmnabmc | --- | 24.375 |
| 531 | ccdmnacbnm | --- | 31.875 |
| 532 | ccdmnacmb | --- | 36.375 |
| 533 | ccdmnambc | --- | 28.875 |
| 534 | ccdmnamcb | --- | 34.875 |
| 535 | ccdmncabm | --- | 37.875 |
| 536 | ccdmncamb | --- | 42.375 |
| 537 | ccdmncmab | --- | 46.875 |
| 538 | ccdmnmabc | --- | 33.375 |
| 539 | ccdmnmacb | --- | 39.375 |
| 540 | ccdmnmcab | --- | 45.375 |
| 541 | ccmabdcnm | --- | 27.375 |
| 542 | ccmabdncm | --- | 21.375 |
| 543 | ccmabdncm | --- | 19.875 |
| 544 | ccmabndcm | --- | 21.375 |
| 545 | ccmabndmc | --- | 19.875 |
| 546 | ccmabnmdc | --- | 24.375 |
| 547 | ccmadbcnm | --- | 27.375 |
| 548 | ccmadbnm | --- | 21.375 |
| 549 | ccmadbnmc | --- | 19.875 |
| 550 | ccmadcbnm | --- | 33.375 |
| 551 | ccmadcnbnm | --- | 33.375 |
| 552 | ccmadcnmb | --- | 37.875 |
| 553 | ccmadnbcnm | --- | 21.375 |

| | | | |
|-----|-----------|-----|--------|
| 554 | ccmadnbmc | --- | 19.875 |
| 555 | ccmadncbm | --- | 27.375 |
| 556 | ccmadncmb | --- | 31.875 |
| 557 | ccmadnmbc | --- | 24.375 |
| 558 | ccmadnmbc | --- | 30.375 |
| 559 | ccmanbdcm | --- | 21.375 |
| 560 | ccmanbdmc | --- | 19.875 |
| 561 | ccmanbmdc | --- | 24.375 |
| 562 | ccmandbcm | --- | 21.375 |
| 563 | ccmandbmc | --- | 19.875 |
| 564 | ccmandcbm | --- | 27.375 |
| 565 | ccmandcmb | --- | 31.875 |
| 566 | ccmandmbc | --- | 24.375 |
| 567 | ccmandmcb | --- | 30.375 |
| 568 | ccmanmbdc | --- | 28.875 |
| 569 | ccmanmdbc | --- | 28.875 |
| 570 | ccmanmdcb | --- | 34.875 |
| 571 | ccmdabcnm | --- | 27.375 |
| 572 | ccmdabncm | --- | 21.375 |
| 573 | ccmdabnmc | --- | 19.875 |
| 574 | ccmdacbnm | --- | 33.375 |
| 575 | ccmdacnbm | --- | 33.375 |
| 576 | ccmdacnmb | --- | 37.875 |
| 577 | ccmdanbcm | --- | 21.375 |
| 578 | ccmdanbmc | --- | 19.875 |
| 579 | ccmdancbm | --- | 27.375 |
| 580 | ccmdancmb | --- | 31.875 |
| 581 | ccmdanmbc | --- | 24.375 |
| 582 | ccmdanmcb | --- | 30.375 |
| 583 | ccmdcabnm | --- | 39.375 |
| 584 | ccmdcanbm | --- | 39.375 |
| 585 | ccmdcanmb | --- | 43.875 |
| 586 | ccmdcnabm | --- | 39.375 |
| 587 | ccmdcnamb | --- | 43.875 |
| 588 | ccmdcnmab | --- | 48.375 |
| 589 | ccmdnabcn | --- | 21.375 |
| 590 | ccmdnabmc | --- | 19.875 |
| 591 | ccmdnacbm | --- | 27.375 |
| 592 | ccmdnacmb | --- | 31.875 |
| 593 | ccmdnambc | --- | 24.375 |
| 594 | ccmdnamcb | --- | 30.375 |
| 595 | ccmdncabm | --- | 33.375 |
| 596 | ccmdncamb | --- | 37.875 |
| 597 | ccmdncmab | --- | 42.375 |
| 598 | ccmdnmabc | --- | 28.875 |
| 599 | ccmdnmacb | --- | 34.875 |
| 600 | ccmdnmcab | --- | 40.875 |
| 601 | ccmnabdcm | --- | 21.375 |
| 602 | ccmnabdmc | --- | 19.875 |
| 603 | ccmnabmdc | --- | 24.375 |
| 604 | ccmnadbcm | --- | 21.375 |
| 605 | ccmnadbmc | --- | 19.875 |
| 606 | ccmnadcbm | --- | 27.375 |
| 607 | ccmnadcmb | --- | 31.875 |
| 608 | ccmnadmcb | --- | 24.375 |
| 609 | ccmnadmcb | --- | 30.375 |

| | | | |
|-----|------------|-----|--------|
| 610 | ccmnambdc | --- | 28.875 |
| 611 | ccmnamdbc | --- | 28.875 |
| 612 | ccmnamdc b | --- | 34.875 |
| 613 | ccmndabc m | --- | 21.375 |
| 614 | ccmndabmc | --- | 19.875 |
| 615 | ccmndacbm | --- | 27.375 |
| 616 | ccmndacmb | --- | 31.875 |
| 617 | ccmndambc | --- | 24.375 |
| 618 | ccmndamcb | --- | 30.375 |
| 619 | ccmndcabm | --- | 33.375 |
| 620 | ccmndcamb | --- | 37.875 |
| 621 | ccmndcmab | --- | 42.375 |
| 622 | ccmndmabc | --- | 28.875 |
| 623 | ccmndmacb | --- | 34.875 |
| 624 | ccmndmcab | --- | 40.875 |
| 625 | ccmnambdc | --- | 33.375 |
| 626 | ccmnmadbc | --- | 33.375 |
| 627 | ccmnmadcb | --- | 39.375 |
| 628 | ccmnmdabc | --- | 33.375 |
| 629 | ccmnmdacb | --- | 39.375 |
| 630 | ccmnmdcab | --- | 45.375 |
| 631 | cmabcdcnm | --- | 22.875 |
| 632 | cmabcdncm | --- | 16.875 |
| 633 | cmabcdnmc | --- | 15.375 |
| 634 | cmabcndcm | --- | 16.875 |
| 635 | cmabcndmc | --- | 15.375 |
| 636 | cmabcnmdc | --- | 19.875 |
| 637 | cmabncdcm | --- | 16.875 |
| 638 | cmabncdmc | --- | 15.375 |
| 639 | cmabncmdc | --- | 19.875 |
| 640 | cmabnmdc | --- | 24.375 |
| 641 | cmacbdcnm | --- | 22.875 |
| 642 | cmacbdncm | --- | 16.875 |
| 643 | cmacbdnmc | --- | 15.375 |
| 644 | cmacbndcm | --- | 16.875 |
| 645 | cmacbndmc | --- | 15.375 |
| 646 | cmacbnmdc | --- | 19.875 |
| 647 | cmacdbcnm | --- | 22.875 |
| 648 | cmacdbnmc | --- | 16.875 |
| 649 | cmacdbnmc | --- | 15.375 |
| 650 | cmacdcbnm | --- | 28.875 |
| 651 | cmacdcnbm | --- | 28.875 |
| 652 | cmacdcnmb | --- | 33.375 |
| 653 | cmacdncm | --- | 16.875 |
| 654 | cmacdncm | --- | 15.375 |
| 655 | cmacdncbm | --- | 22.875 |
| 656 | cmacdncmb | --- | 27.375 |
| 657 | cmacdncmb | --- | 19.875 |
| 658 | cmacdncmb | --- | 25.875 |
| 659 | cmacnbdcm | --- | 16.875 |
| 660 | cmacnbdmc | --- | 15.375 |
| 661 | cmacnbmdc | --- | 19.875 |
| 662 | cmacndbcm | --- | 16.875 |
| 663 | cmacndbmc | --- | 15.375 |
| 664 | cmacndcbm | --- | 22.875 |
| 665 | cmacndcmb | --- | 27.375 |

| | | | |
|-----|-----------|-----|--------|
| 666 | cmacndmbc | --- | 19.875 |
| 667 | cmacndmcb | --- | 25.875 |
| 668 | cmacnmbdc | --- | 24.375 |
| 669 | cmacnmdbc | --- | 24.375 |
| 670 | cmacnmdbc | --- | 30.375 |
| 671 | cmancbdc | --- | 16.875 |
| 672 | cmancbdc | --- | 15.375 |
| 673 | cmancbdc | --- | 19.875 |
| 674 | cmancbdc | --- | 24.375 |
| 675 | cmancbdc | --- | 16.875 |
| 676 | cmancbdc | --- | 15.375 |
| 677 | cmancbdc | --- | 19.875 |
| 678 | cmancbdc | --- | 16.875 |
| 679 | cmancbdc | --- | 15.375 |
| 680 | cmancbdc | --- | 22.875 |
| 681 | cmancbdc | --- | 27.375 |
| 682 | cmancbdc | --- | 19.875 |
| 683 | cmancbdc | --- | 25.875 |
| 684 | cmancbdc | --- | 24.375 |
| 685 | cmancbdc | --- | 24.375 |
| 686 | cmancbdc | --- | 30.375 |
| 687 | cmancbdc | --- | 28.875 |
| 688 | cmancbdc | --- | 28.875 |
| 689 | cmancbdc | --- | 28.875 |
| 690 | cmancbdc | --- | 34.875 |
| 691 | cmcabdc | --- | 22.875 |
| 692 | cmcabdc | --- | 16.875 |
| 693 | cmcabdc | --- | 15.375 |
| 694 | cmcabdc | --- | 16.875 |
| 695 | cmcabdc | --- | 15.375 |
| 696 | cmcabdc | --- | 19.875 |
| 697 | cmcabdc | --- | 22.875 |
| 698 | cmcabdc | --- | 16.875 |
| 699 | cmcabdc | --- | 15.375 |
| 700 | cmcabdc | --- | 28.875 |
| 701 | cmcabdc | --- | 28.875 |
| 702 | cmcabdc | --- | 33.375 |
| 703 | cmcabdc | --- | 16.875 |
| 704 | cmcabdc | --- | 15.375 |
| 705 | cmcabdc | --- | 22.875 |
| 706 | cmcabdc | --- | 27.375 |
| 707 | cmcabdc | --- | 19.875 |
| 708 | cmcabdc | --- | 25.875 |
| 709 | cmcabdc | --- | 16.875 |
| 710 | cmcabdc | --- | 15.375 |
| 711 | cmcabdc | --- | 19.875 |
| 712 | cmcabdc | --- | 16.875 |
| 713 | cmcabdc | --- | 15.375 |
| 714 | cmcabdc | --- | 22.875 |
| 715 | cmcabdc | --- | 27.375 |
| 716 | cmcabdc | --- | 19.875 |
| 717 | cmcabdc | --- | 25.875 |
| 718 | cmcabdc | --- | 24.375 |
| 719 | cmcabdc | --- | 24.375 |
| 720 | cmcabdc | --- | 30.375 |
| 721 | cmcabdc | --- | 22.875 |

| | | | |
|-----|------------|-----|--------|
| 722 | cmcdabncm | --- | 16.875 |
| 723 | cmcdabnmc | --- | 15.375 |
| 724 | cmcdacbnm | --- | 28.875 |
| 725 | cmcdacnbm | --- | 28.875 |
| 726 | cmcdacnmb | --- | 33.375 |
| 727 | cmcdanbcm | --- | 16.875 |
| 728 | cmcdanbmc | --- | 15.375 |
| 729 | cmcdancbm | --- | 22.875 |
| 730 | cmcdancmb | --- | 27.375 |
| 731 | cmcdanmbc | --- | 19.875 |
| 732 | cmcdanmcb | --- | 25.875 |
| 733 | cmcdcabnm | --- | 34.875 |
| 734 | cmcdcanbm | --- | 34.875 |
| 735 | cmcdcanmb | --- | 39.375 |
| 736 | cmcdcnabm | --- | 34.875 |
| 737 | cmcdcnamb | --- | 39.375 |
| 738 | cmcdcnmab | --- | 43.875 |
| 739 | cmcdnabcm | --- | 16.875 |
| 740 | cmcdnabmc | --- | 15.375 |
| 741 | cmcdnacbm | --- | 22.875 |
| 742 | cmcdnacmb | --- | 27.375 |
| 743 | cmcdnambc | --- | 19.875 |
| 744 | cmcdnamcb | --- | 25.875 |
| 745 | cmcdncabm | --- | 28.875 |
| 746 | cmcdncamb | --- | 33.375 |
| 747 | cmcdncmab | --- | 37.875 |
| 748 | cmcdnmabc | --- | 24.375 |
| 749 | cmcdnmacb | --- | 30.375 |
| 750 | cmcdnmcab | --- | 36.375 |
| 751 | cmcnabdcm | --- | 16.875 |
| 752 | cmcnabdmc | --- | 15.375 |
| 753 | cmcnabmdc | --- | 19.875 |
| 754 | cmcnadbcm | --- | 16.875 |
| 755 | cmcnadbmc | --- | 15.375 |
| 756 | cmcnadcbm | --- | 22.875 |
| 757 | cmcnadcmb | --- | 27.375 |
| 758 | cmcnadmbc | --- | 19.875 |
| 759 | cmcnadmcb | --- | 25.875 |
| 760 | cmcnambdc | --- | 24.375 |
| 761 | cmcnamdbc | --- | 24.375 |
| 762 | cmcnamdc b | --- | 30.375 |
| 763 | cmcndabcm | --- | 16.875 |
| 764 | cmcndabmc | --- | 15.375 |
| 765 | cmcndacbm | --- | 22.875 |
| 766 | cmcndacmb | --- | 27.375 |
| 767 | cmcndambc | --- | 19.875 |
| 768 | cmcndamcb | --- | 25.875 |
| 769 | cmcndcabm | --- | 28.875 |
| 770 | cmcndcamb | --- | 33.375 |
| 771 | cmcndcmab | --- | 37.875 |
| 772 | cmcndmabc | --- | 24.375 |
| 773 | cmcndmacb | --- | 30.375 |
| 774 | cmcndmcab | --- | 36.375 |
| 775 | cmcnmabdc | --- | 28.875 |
| 776 | cmcnmadbc | --- | 28.875 |
| 777 | cmcnmadcb | --- | 34.875 |

| | | | |
|-----|------------|-----|--------|
| 778 | cmcnmdabc | --- | 28.875 |
| 779 | cmcnmdacb | --- | 34.875 |
| 780 | cmcnmdcab | --- | 40.875 |
| 781 | cmnabcdcm | --- | 16.875 |
| 782 | cmnabcdmc | --- | 15.375 |
| 783 | cmnabcmdc | --- | 19.875 |
| 784 | cmnabmcdc | --- | 24.375 |
| 785 | cmnacbcdcm | --- | 16.875 |
| 786 | cmnacbdmc | --- | 15.375 |
| 787 | cmnacbmcdc | --- | 19.875 |
| 788 | cmnacdbcm | --- | 16.875 |
| 789 | cmnacdbmc | --- | 15.375 |
| 790 | cmnacdcbm | --- | 22.875 |
| 791 | cmnacdcmb | --- | 27.375 |
| 792 | cmnacdmcb | --- | 19.875 |
| 793 | cmnacdmcb | --- | 25.875 |
| 794 | cmnacmbdc | --- | 24.375 |
| 795 | cmnacmdbc | --- | 24.375 |
| 796 | cmnacmdcb | --- | 30.375 |
| 797 | cmnambcdc | --- | 28.875 |
| 798 | cmnamcbdc | --- | 28.875 |
| 799 | cmnamcdbc | --- | 28.875 |
| 800 | cmnamcdcb | --- | 34.875 |
| 801 | cmncabdc | --- | 16.875 |
| 802 | cmncabdmc | --- | 15.375 |
| 803 | cmncabmcdc | --- | 19.875 |
| 804 | cmncadbcm | --- | 16.875 |
| 805 | cmncadbmc | --- | 15.375 |
| 806 | cmncadcmb | --- | 22.875 |
| 807 | cmncadcmb | --- | 27.375 |
| 808 | cmncadmcb | --- | 19.875 |
| 809 | cmncadmcb | --- | 25.875 |
| 810 | cmncambdc | --- | 24.375 |
| 811 | cmncamdbc | --- | 24.375 |
| 812 | cmncamdcb | --- | 30.375 |
| 813 | cmncdabcm | --- | 16.875 |
| 814 | cmncdabmc | --- | 15.375 |
| 815 | cmncdacbm | --- | 22.875 |
| 816 | cmncdacmb | --- | 27.375 |
| 817 | cmncdambc | --- | 19.875 |
| 818 | cmncdamcb | --- | 25.875 |
| 819 | cmncdcabm | --- | 28.875 |
| 820 | cmncdcamb | --- | 33.375 |
| 821 | cmncdcamb | --- | 37.875 |
| 822 | cmncdmabc | --- | 24.375 |
| 823 | cmncdmacb | --- | 30.375 |
| 824 | cmncdmcab | --- | 36.375 |
| 825 | cmncmabdc | --- | 28.875 |
| 826 | cmncmadbc | --- | 28.875 |
| 827 | cmncmadcb | --- | 34.875 |
| 828 | cmncmdabc | --- | 28.875 |
| 829 | cmncmdacb | --- | 34.875 |
| 830 | cmncmdcab | --- | 40.875 |
| 831 | cmnmabcdc | --- | 33.375 |
| 832 | cmnmacbdc | --- | 33.375 |
| 833 | cmnmacdbc | --- | 33.375 |

| | | | |
|-----|------------|-----|--------|
| 834 | cmnmacdcb | --- | 39.375 |
| 835 | cmnmcabdc | --- | 33.375 |
| 836 | cmnmcadb | --- | 33.375 |
| 837 | cmnmcadcb | --- | 39.375 |
| 838 | cmnmcadbc | --- | 33.375 |
| 839 | cmnmcadcb | --- | 39.375 |
| 840 | cmnmcadcb | --- | 45.375 |
| 841 | mabccdcnm | --- | 36.375 |
| 842 | mabccdcnm | --- | 30.375 |
| 843 | mabccdnmc | --- | 28.875 |
| 844 | mabccndcm | --- | 30.375 |
| 845 | mabccndmc | --- | 28.875 |
| 846 | mabccnmdc | --- | 33.375 |
| 847 | mabcncdcm | --- | 30.375 |
| 848 | mabcncdmc | --- | 28.875 |
| 849 | mabcncmdc | --- | 33.375 |
| 850 | mabcnmdc | --- | 37.875 |
| 851 | mabnccdcn | --- | 36.375 |
| 852 | mabnccdcn | --- | 34.875 |
| 853 | mabnccmdc | --- | 39.375 |
| 854 | mabnccmdc | --- | 43.875 |
| 855 | mabnccmdc | --- | 54.375 |
| 856 | macbcdcnm | --- | 30.375 |
| 857 | macbcdcnm | --- | 24.375 |
| 858 | macbcdnmc | --- | 22.875 |
| 859 | macbcdnmc | --- | 24.375 |
| 860 | macbcdnmc | --- | 22.875 |
| 861 | macbcnmdc | --- | 27.375 |
| 862 | macbcndcm | --- | 24.375 |
| 863 | macbcndmc | --- | 22.875 |
| 864 | macbcnmdc | --- | 27.375 |
| 865 | macbcnmdc | --- | 31.875 |
| 866 | maccbdcnm | --- | 30.375 |
| 867 | maccbdcnm | --- | 24.375 |
| 868 | maccbdncm | --- | 22.875 |
| 869 | maccbdndcm | --- | 24.375 |
| 870 | maccbdndmc | --- | 22.875 |
| 871 | maccbnmdc | --- | 27.375 |
| 872 | maccdbenm | --- | 30.375 |
| 873 | maccdbncm | --- | 24.375 |
| 874 | maccdbnmc | --- | 22.875 |
| 875 | maccdcbnm | --- | 36.375 |
| 876 | maccdcnbm | --- | 36.375 |
| 877 | maccdcnmb | --- | 40.875 |
| 878 | maccdnbcm | --- | 24.375 |
| 879 | maccdnbmc | --- | 22.875 |
| 880 | maccdncbm | --- | 30.375 |
| 881 | maccdncmb | --- | 34.875 |
| 882 | maccdnmbc | --- | 27.375 |
| 883 | maccdnmc | --- | 33.375 |
| 884 | maccnbdcm | --- | 24.375 |
| 885 | maccnbdmc | --- | 22.875 |
| 886 | maccnbmdc | --- | 27.375 |
| 887 | maccnbdcm | --- | 24.375 |
| 888 | maccnbdmc | --- | 22.875 |
| 889 | maccnbdcm | --- | 30.375 |

| | | | |
|-----|------------|-----|--------|
| 890 | maccndcmb | --- | 34.875 |
| 891 | maccndmcb | --- | 27.375 |
| 892 | maccndmcb | --- | 33.375 |
| 893 | maccnmbdc | --- | 31.875 |
| 894 | maccnmdbc | --- | 31.875 |
| 895 | maccnmdcb | --- | 37.875 |
| 896 | macnbcddcm | --- | 24.375 |
| 897 | macnbcddmc | --- | 22.875 |
| 898 | macnbcddc | --- | 27.375 |
| 899 | macnbmddc | --- | 31.875 |
| 900 | macnbcddcm | --- | 24.375 |
| 901 | macnbcddmc | --- | 22.875 |
| 902 | macnbcddc | --- | 27.375 |
| 903 | macncdbcm | --- | 24.375 |
| 904 | macncdbmc | --- | 22.875 |
| 905 | macncdcbm | --- | 30.375 |
| 906 | macncdcmb | --- | 34.875 |
| 907 | macncdmbc | --- | 27.375 |
| 908 | macncdmcb | --- | 33.375 |
| 909 | macncmbdc | --- | 31.875 |
| 910 | macncmdbc | --- | 31.875 |
| 911 | macncmdcb | --- | 37.875 |
| 912 | macnmbcdc | --- | 36.375 |
| 913 | macnmbcdc | --- | 36.375 |
| 914 | macnmcdbc | --- | 36.375 |
| 915 | macnmcddb | --- | 42.375 |
| 916 | manbccddcm | --- | 36.375 |
| 917 | manbccddmc | --- | 34.875 |
| 918 | manbccddc | --- | 39.375 |
| 919 | manbcmddc | --- | 43.875 |
| 920 | manbmccdc | --- | 54.375 |
| 921 | mancbddcm | --- | 30.375 |
| 922 | mancbddmc | --- | 28.875 |
| 923 | mancbddc | --- | 33.375 |
| 924 | mancbmddc | --- | 37.875 |
| 925 | manccbddcm | --- | 30.375 |
| 926 | manccbddmc | --- | 28.875 |
| 927 | manccbddc | --- | 33.375 |
| 928 | manccdbcm | --- | 30.375 |
| 929 | manccdbmc | --- | 28.875 |
| 930 | manccdcbm | --- | 36.375 |
| 931 | manccdcmb | --- | 40.875 |
| 932 | manccdmcb | --- | 33.375 |
| 933 | manccdmcb | --- | 39.375 |
| 934 | manccmbdc | --- | 37.875 |
| 935 | manccmdbc | --- | 37.875 |
| 936 | manccmdcb | --- | 43.875 |
| 937 | mancmbcdc | --- | 42.375 |
| 938 | mancmbcdc | --- | 42.375 |
| 939 | mancmddc | --- | 42.375 |
| 940 | mancmddb | --- | 48.375 |
| 941 | manmbccdc | --- | 58.875 |
| 942 | manmbcdc | --- | 52.875 |
| 943 | manmccbdc | --- | 52.875 |
| 944 | manmccdbc | --- | 52.875 |
| 945 | manmccddb | --- | 58.875 |

| | | | |
|------|------------|-----|--------|
| 946 | mcabcdcnm | --- | 24.375 |
| 947 | mcabcdncm | --- | 18.375 |
| 948 | mcabcdnmc | --- | 16.875 |
| 949 | mcabcndcm | --- | 18.375 |
| 950 | mcabcndmc | --- | 16.875 |
| 951 | mcabcnmdc | --- | 21.375 |
| 952 | mcabncdcm | --- | 18.375 |
| 953 | mcabncdmc | --- | 16.875 |
| 954 | mcabncmdc | --- | 21.375 |
| 955 | mcabnmdc | --- | 25.875 |
| 956 | mcacbdcnm | --- | 24.375 |
| 957 | mcacbdncm | --- | 18.375 |
| 958 | mcacbdnmc | --- | 16.875 |
| 959 | mcacbndcm | --- | 18.375 |
| 960 | mcacbndmc | --- | 16.875 |
| 961 | mcacbnmdc | --- | 21.375 |
| 962 | mcacdbcnm | --- | 24.375 |
| 963 | mcacdbncm | --- | 18.375 |
| 964 | mcacdbnmc | --- | 16.875 |
| 965 | mcacdcbnm | --- | 30.375 |
| 966 | mcacdcnbm | --- | 30.375 |
| 967 | mcacdcnmb | --- | 34.875 |
| 968 | mcacdncbm | --- | 18.375 |
| 969 | mcacdncmc | --- | 16.875 |
| 970 | mcacdncmb | --- | 24.375 |
| 971 | mcacdncmb | --- | 28.875 |
| 972 | mcacdncmb | --- | 21.375 |
| 973 | mcacdncmb | --- | 27.375 |
| 974 | mcacnbdcm | --- | 18.375 |
| 975 | mcacnbdmc | --- | 16.875 |
| 976 | mcacnbmdc | --- | 21.375 |
| 977 | mcacndbcm | --- | 18.375 |
| 978 | mcacndbmc | --- | 16.875 |
| 979 | mcacndcbm | --- | 24.375 |
| 980 | mcacndcmb | --- | 28.875 |
| 981 | mcacndmcb | --- | 21.375 |
| 982 | mcacndmcb | --- | 27.375 |
| 983 | mcacnmbdc | --- | 25.875 |
| 984 | mcacnmdbc | --- | 25.875 |
| 985 | mcacnmdcb | --- | 31.875 |
| 986 | mcanbcdcm | --- | 18.375 |
| 987 | mcanbcdmc | --- | 16.875 |
| 988 | mcanbcmdc | --- | 21.375 |
| 989 | mcanbmdc | --- | 25.875 |
| 990 | mcancbdcm | --- | 18.375 |
| 991 | mcancbdmc | --- | 16.875 |
| 992 | mcancbmdc | --- | 21.375 |
| 993 | mcancdbcm | --- | 18.375 |
| 994 | mcancdbmc | --- | 16.875 |
| 995 | mcancdcblm | --- | 24.375 |
| 996 | mcancdcmb | --- | 28.875 |
| 997 | mcancdmcb | --- | 21.375 |
| 998 | mcancdmcb | --- | 27.375 |
| 999 | mcancmbdc | --- | 25.875 |
| 1000 | mcancmdbc | --- | 25.875 |
| 1001 | mcancmdcb | --- | 31.875 |

| | | | |
|------|-----------|-----|--------|
| 1002 | mcanmbcdc | --- | 30.375 |
| 1003 | mcanmcbdc | --- | 30.375 |
| 1004 | mcanmcdbc | --- | 30.375 |
| 1005 | mcanmcdcb | --- | 36.375 |
| 1006 | mccabdcnm | --- | 24.375 |
| 1007 | mccabdncm | --- | 18.375 |
| 1008 | mccabdncm | --- | 16.875 |
| 1009 | mccabndcm | --- | 18.375 |
| 1010 | mccabndmc | --- | 16.875 |
| 1011 | mccabnmdc | --- | 21.375 |
| 1012 | mccadbcnm | --- | 24.375 |
| 1013 | mccadbncm | --- | 18.375 |
| 1014 | mccadbncm | --- | 16.875 |
| 1015 | mccadcbnm | --- | 30.375 |
| 1016 | mccadcncm | --- | 30.375 |
| 1017 | mccadcncm | --- | 34.875 |
| 1018 | mccadnbcn | --- | 18.375 |
| 1019 | mccadnbcn | --- | 16.875 |
| 1020 | mccadncbn | --- | 24.375 |
| 1021 | mccadncbn | --- | 28.875 |
| 1022 | mccadncbn | --- | 21.375 |
| 1023 | mccadncbn | --- | 27.375 |
| 1024 | mccanbdcn | --- | 18.375 |
| 1025 | mccanbdcn | --- | 16.875 |
| 1026 | mccanbmdc | --- | 21.375 |
| 1027 | mccandbcn | --- | 18.375 |
| 1028 | mccandbmc | --- | 16.875 |
| 1029 | mccandcbn | --- | 24.375 |
| 1030 | mccandcmb | --- | 28.875 |
| 1031 | mccandmcb | --- | 21.375 |
| 1032 | mccandmcb | --- | 27.375 |
| 1033 | mccanmbdc | --- | 25.875 |
| 1034 | mccanmbdc | --- | 25.875 |
| 1035 | mccanmdbc | --- | 31.875 |
| 1036 | mccdabcnm | --- | 24.375 |
| 1037 | mccdabncm | --- | 18.375 |
| 1038 | mccdabnmc | --- | 16.875 |
| 1039 | mccdacbnm | --- | 30.375 |
| 1040 | mccdacnbn | --- | 30.375 |
| 1041 | mccdacnbn | --- | 34.875 |
| 1042 | mccdancbn | --- | 18.375 |
| 1043 | mccdancbn | --- | 16.875 |
| 1044 | mccdancbn | --- | 24.375 |
| 1045 | mccdancbn | --- | 28.875 |
| 1046 | mccdancbn | --- | 21.375 |
| 1047 | mccdancbn | --- | 27.375 |
| 1048 | mccdcbnbn | --- | 36.375 |
| 1049 | mccdcanbn | --- | 36.375 |
| 1050 | mccdcanbn | --- | 40.875 |
| 1051 | mccdcnabn | --- | 36.375 |
| 1052 | mccdcnamb | --- | 40.875 |
| 1053 | mccdcnmab | --- | 45.375 |
| 1054 | mccdncbnm | --- | 18.375 |
| 1055 | mccdncbnm | --- | 16.875 |
| 1056 | mccdncbnm | --- | 24.375 |
| 1057 | mccdncbnm | --- | 28.875 |

| | | | |
|------|-----------|-----|--------|
| 1058 | mccdnambc | --- | 21.375 |
| 1059 | mccdnamcb | --- | 27.375 |
| 1060 | mccdncabm | --- | 30.375 |
| 1061 | mccdncamb | --- | 34.875 |
| 1062 | mccdncmab | --- | 39.375 |
| 1063 | mccdnmabc | --- | 25.875 |
| 1064 | mccdnmacb | --- | 31.875 |
| 1065 | mccdnmcab | --- | 37.875 |
| 1066 | mccnabdcn | --- | 18.375 |
| 1067 | mccnabdmc | --- | 16.875 |
| 1068 | mccnabmdc | --- | 21.375 |
| 1069 | mccnadbcm | --- | 18.375 |
| 1070 | mccnadbmc | --- | 16.875 |
| 1071 | mccnadcbm | --- | 24.375 |
| 1072 | mccnadcmb | --- | 28.875 |
| 1073 | mccnadmbc | --- | 21.375 |
| 1074 | mccnadmcb | --- | 27.375 |
| 1075 | mccnamdbc | --- | 25.875 |
| 1076 | mccnamdbc | --- | 25.875 |
| 1077 | mccnamdcb | --- | 31.875 |
| 1078 | mccndabcm | --- | 18.375 |
| 1079 | mccndabmc | --- | 16.875 |
| 1080 | mccndacbm | --- | 24.375 |
| 1081 | mccndacmb | --- | 28.875 |
| 1082 | mccndambc | --- | 21.375 |
| 1083 | mccndamcb | --- | 27.375 |
| 1084 | mccndcabm | --- | 30.375 |
| 1085 | mccndcamb | --- | 34.875 |
| 1086 | mccndcmab | --- | 39.375 |
| 1087 | mccndmabc | --- | 25.875 |
| 1088 | mccndmacb | --- | 31.875 |
| 1089 | mccndmcab | --- | 37.875 |
| 1090 | mccnmabdc | --- | 30.375 |
| 1091 | mccnmadbc | --- | 30.375 |
| 1092 | mccnmadcb | --- | 36.375 |
| 1093 | mccnmdabc | --- | 30.375 |
| 1094 | mccnmdacb | --- | 36.375 |
| 1095 | mccnmdcab | --- | 42.375 |
| 1096 | mcnabcdcm | --- | 18.375 |
| 1097 | mcnabcdmc | --- | 16.875 |
| 1098 | mcnabcmdc | --- | 21.375 |
| 1099 | mcnabmcde | --- | 25.875 |
| 1100 | mcnacbdcm | --- | 18.375 |
| 1101 | mcnacbdmc | --- | 16.875 |
| 1102 | mcnacbmdc | --- | 21.375 |
| 1103 | mcnacdbcm | --- | 18.375 |
| 1104 | mcnacdbmc | --- | 16.875 |
| 1105 | mcnacdbcm | --- | 24.375 |
| 1106 | mcnacdcmb | --- | 28.875 |
| 1107 | mcnacdmcb | --- | 21.375 |
| 1108 | mcnacdmcb | --- | 27.375 |
| 1109 | mcnacmbdc | --- | 25.875 |
| 1110 | mcnacmdbc | --- | 25.875 |
| 1111 | mcnacmdcb | --- | 31.875 |
| 1112 | mcnambedc | --- | 30.375 |
| 1113 | mcnamcbdc | --- | 30.375 |

| | | | |
|------|------------|-----|--------|
| 1114 | mcnamcdbc | --- | 30.375 |
| 1115 | mcnamcdb | --- | 36.375 |
| 1116 | mencabdem | --- | 18.375 |
| 1117 | mencabdmc | --- | 16.875 |
| 1118 | mencabmdc | --- | 21.375 |
| 1119 | mencadbcm | --- | 18.375 |
| 1120 | mencadbmc | --- | 16.875 |
| 1121 | mencadcbm | --- | 24.375 |
| 1122 | mencadcmb | --- | 28.875 |
| 1123 | mencadmbc | --- | 21.375 |
| 1124 | mencadmcb | --- | 27.375 |
| 1125 | mencambdc | --- | 25.875 |
| 1126 | mencamdbc | --- | 25.875 |
| 1127 | mencamdc | --- | 31.875 |
| 1128 | mencdabcm | --- | 18.375 |
| 1129 | mencdabmc | --- | 16.875 |
| 1130 | mencdacbm | --- | 24.375 |
| 1131 | mencdacmb | --- | 28.875 |
| 1132 | mencdambc | --- | 21.375 |
| 1133 | mencdamcb | --- | 27.375 |
| 1134 | mencedcabm | --- | 30.375 |
| 1135 | mencedcamb | --- | 34.875 |
| 1136 | mencedcmab | --- | 39.375 |
| 1137 | mencedmabc | --- | 25.875 |
| 1138 | mencedmacb | --- | 31.875 |
| 1139 | mencedmcab | --- | 37.875 |
| 1140 | mencmabdc | --- | 30.375 |
| 1141 | mencmadbc | --- | 30.375 |
| 1142 | mencmadcb | --- | 36.375 |
| 1143 | mencmdabc | --- | 30.375 |
| 1144 | mencmdacb | --- | 36.375 |
| 1145 | mencmdcab | --- | 42.375 |
| 1146 | mcnmabcde | --- | 34.875 |
| 1147 | mcnmacbdc | --- | 34.875 |
| 1148 | mcnmacdbc | --- | 34.875 |
| 1149 | mcnmacdcb | --- | 40.875 |
| 1150 | mcnmcabdc | --- | 34.875 |
| 1151 | mcnmcadb | --- | 34.875 |
| 1152 | mcnmcadb | --- | 40.875 |
| 1153 | mcnmcadb | --- | 34.875 |
| 1154 | mcnmcadb | --- | 40.875 |
| 1155 | mcnmcadb | --- | 46.875 |
| 1156 | mnabccdem | --- | 36.375 |
| 1157 | mnabccdm | --- | 34.875 |
| 1158 | mnabccmd | --- | 39.375 |
| 1159 | mnabcmcdc | --- | 43.875 |
| 1160 | mnabmccdc | --- | 54.375 |
| 1161 | mnacbedem | --- | 30.375 |
| 1162 | mnacbedm | --- | 28.875 |
| 1163 | mnacbcmcd | --- | 33.375 |
| 1164 | mnacbmcdc | --- | 37.875 |
| 1165 | mnaccbdem | --- | 30.375 |
| 1166 | mnaccbdm | --- | 28.875 |
| 1167 | mnaccbmd | --- | 33.375 |
| 1168 | mnaccdbem | --- | 30.375 |
| 1169 | mnaccdbm | --- | 28.875 |

| | | | |
|------|-------------|-----|--------|
| 1170 | mnaccdcbm | --- | 36.375 |
| 1171 | mnaccdcmb | --- | 40.875 |
| 1172 | mnaccdmbe | --- | 33.375 |
| 1173 | mnaccdmcb | --- | 39.375 |
| 1174 | mnaccmbdc | --- | 37.875 |
| 1175 | mnaccmdbc | --- | 37.875 |
| 1176 | mnaccmdeb | --- | 43.875 |
| 1177 | mnacmbcdc | --- | 42.375 |
| 1178 | mnacmbdc | --- | 42.375 |
| 1179 | mnacmdbc | --- | 42.375 |
| 1180 | mnacmdeb | --- | 48.375 |
| 1181 | mnambccdc | --- | 58.875 |
| 1182 | mnambcdc | --- | 52.875 |
| 1183 | mnamccbdc | --- | 52.875 |
| 1184 | mnamccdbc | --- | 52.875 |
| 1185 | mnamccdeb | --- | 58.875 |
| 1186 | mncabcedm | --- | 24.375 |
| 1187 | mncabcedmc | --- | 22.875 |
| 1188 | mncabcmde | --- | 27.375 |
| 1189 | mncabmcdc | --- | 31.875 |
| 1190 | mncacbedm | --- | 24.375 |
| 1191 | mncacbedmc | --- | 22.875 |
| 1192 | mncacbmde | --- | 27.375 |
| 1193 | mncacdbcm | --- | 24.375 |
| 1194 | mncacdbmc | --- | 22.875 |
| 1195 | mncacdcbm | --- | 30.375 |
| 1196 | mncacdcmb | --- | 34.875 |
| 1197 | mncacdmbc | --- | 27.375 |
| 1198 | mncacdmb | --- | 33.375 |
| 1199 | mncacmbdc | --- | 31.875 |
| 1200 | mncacmdbc | --- | 31.875 |
| 1201 | mncacmdeb | --- | 37.875 |
| 1202 | mncambcdc | --- | 36.375 |
| 1203 | mncambdc | --- | 36.375 |
| 1204 | mncamcdc | --- | 36.375 |
| 1205 | mncamdeb | --- | 42.375 |
| 1206 | mnccabcedm | --- | 24.375 |
| 1207 | mnccabcedmc | --- | 22.875 |
| 1208 | mnccabcmde | --- | 27.375 |
| 1209 | mnccadbcm | --- | 24.375 |
| 1210 | mnccadbmc | --- | 22.875 |
| 1211 | mnccadcbm | --- | 30.375 |
| 1212 | mnccadcmb | --- | 34.875 |
| 1213 | mnccadmbe | --- | 27.375 |
| 1214 | mnccadmcb | --- | 33.375 |
| 1215 | mnccambdc | --- | 31.875 |
| 1216 | mnccamdbc | --- | 31.875 |
| 1217 | mnccamdeb | --- | 37.875 |
| 1218 | mnccdabcm | --- | 24.375 |
| 1219 | mnccdabmc | --- | 22.875 |
| 1220 | mnccdacbm | --- | 30.375 |
| 1221 | mnccdacmb | --- | 34.875 |
| 1222 | mnccdambc | --- | 27.375 |
| 1223 | mnccdamb | --- | 33.375 |
| 1224 | mnccdcabm | --- | 36.375 |
| 1225 | mnccdcamb | --- | 40.875 |

| | | | |
|------|-----------|-----|--------|
| 1226 | mnccdcab | --- | 45.375 |
| 1227 | mnccdmabc | --- | 31.875 |
| 1228 | mnccdmacb | --- | 37.875 |
| 1229 | mnccdmcab | --- | 43.875 |
| 1230 | mnccmabdc | --- | 36.375 |
| 1231 | mnccmadbc | --- | 36.375 |
| 1232 | mnccmadcb | --- | 42.375 |
| 1233 | mnccmdabc | --- | 36.375 |
| 1234 | mnccmdacb | --- | 42.375 |
| 1235 | mnccmdcab | --- | 48.375 |
| 1236 | mncmabcde | --- | 40.875 |
| 1237 | mncmacbde | --- | 40.875 |
| 1238 | mncmacdbc | --- | 40.875 |
| 1239 | mncmacdcb | --- | 46.875 |
| 1240 | mncmcabde | --- | 40.875 |
| 1241 | mncmcadbc | --- | 40.875 |
| 1242 | mncmcadcb | --- | 46.875 |
| 1243 | mncmcdabc | --- | 40.875 |
| 1244 | mncmcdacb | --- | 46.875 |
| 1245 | mncmcdcab | --- | 52.875 |
| 1246 | mnmacbcde | --- | 63.375 |
| 1247 | mnmacbcde | --- | 57.375 |
| 1248 | mnmaccbde | --- | 57.375 |
| 1249 | mnmaccbdc | --- | 57.375 |
| 1250 | mnmacdcb | --- | 63.375 |
| 1251 | mnmacbcde | --- | 51.375 |
| 1252 | mnmacbcde | --- | 51.375 |
| 1253 | mnmacdbc | --- | 51.375 |
| 1254 | mnmacdcb | --- | 57.375 |
| 1255 | mnmcabde | --- | 51.375 |
| 1256 | mnmcadbc | --- | 51.375 |
| 1257 | mnmcadcb | --- | 57.375 |
| 1258 | mnmcadabc | --- | 51.375 |
| 1259 | mnmcadacb | --- | 57.375 |
| 1260 | mnmcadcab | --- | 63.375 |

---Min Value---
Minimal Cost-Value: 15.375

7. Minimal Possible Schedules

S.No. Possible Sequence Cost-Value

| | | | |
|---|-----------|-----|--------|
| 1 | cmabcdnmc | --- | 15.375 |
| 2 | cmabcndmc | --- | 15.375 |
| 3 | cmabncdmc | --- | 15.375 |
| 4 | cmacbdnmc | --- | 15.375 |
| 5 | cmacbndmc | --- | 15.375 |
| 6 | cmacdbnmc | --- | 15.375 |
| 7 | cmacdnbmc | --- | 15.375 |
| 8 | cmacnbdmc | --- | 15.375 |

| | | | |
|----------|-------------------------|------------|---------------|
| 9 | <i>cmacndbmc</i> | --- | 15.375 |
| 10 | cmanbcdmc | --- | 15.375 |
| 11 | cmancbdmc | --- | 15.375 |
| 12 | cmancdbmc | --- | 15.375 |
| 13 | cmcabdnmc | --- | 15.375 |
| 14 | cmcabndmc | --- | 15.375 |
| 15 | cmcadbnmc | --- | 15.375 |
| 16 | cmcadnbmc | --- | 15.375 |
| 17 | cmcanbdmc | --- | 15.375 |
| 18 | cmcandbmc | --- | 15.375 |
| 19 | cmcdabnmc | --- | 15.375 |
| 20 | cmcdanbmc | --- | 15.375 |
| 21 | cmcdnabmc | --- | 15.375 |
| 22 | cmcnabdmc | --- | 15.375 |
| 23 | cmcnadbmc | --- | 15.375 |
| 24 | cmcndabmc | --- | 15.375 |
| 25 | cmnabcdmc | --- | 15.375 |
| 26 | cmnacbdmc | --- | 15.375 |
| 27 | cmnacdbmc | --- | 15.375 |
| 28 | cmncabdmc | --- | 15.375 |
| 29 | cmncadbmc | --- | 15.375 |
| 30 | cmncdabmc | --- | 15.375 |

Output:

cmacndbmc is the schedule generated by Proposed -EDD is found in Possible sequence list(Feasibility case).Moreover, this sequence is also found in list of minimal cost sequence(Optimal case). Hence, it is shown empirically that Proposed -EDD is both feasible and optimal.

Chapter 7

CONCLUSION AND FUTURE RECOMMENDATION

Flexible assembly lines that have negligible switch-over costs from one product to another make it possible to implement flexible JIT production, which requires producing only the necessary products in the necessary quantities at necessary times. A JIT system being a pull system initiates any supplying process only if there is another process that requires the supplying process output (subassembly, part, raw material). As a result, it is the final assembly which is the focus for scheduling. The problem of determining a sequence of final assembly such that the quantity of each part used in the assembly process is kept as close to constant as possible throughout the working time which is known as balancing the schedule. Our concern in this dissertation, however, is to extract the best schedule from the possible schedule.

In this dissertation, under the constraint that none of the chains are overlapping, it is shown that by considering each chain as a pseudo job and their length as a demands, we can have a pseudo schedule from EDD, which is later replaced by the real job, can lead a combined chain sequences which is both feasible and optimal (shown practically).

Still a lot of questions are left open. It is let to identify either we can establish the mathematical derivations of Proposed EDD's feasibility and optimality. It is also remain open either we can achieve a similar algorithm for overlapping sequences. Moreover, it is also let open that a similar achievement can be obtained with other algorithms like Cost Assignment, Nearest Integer Point, Dynamic Programming, and so on.

REFERENCES

1. Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, C. and Weglarz, J., "Scheduling computer and manufacturing processes", Springer, Berlin (1996).
2. Brouner, N. and Crama, Y., "The maximum deviation just-in-time scheduling problem", *Discrete Applied Mathematics* 134 (2004) 25-50.
3. Brucker, P., "Scheduling Algorithms", Springer, Verlag 2 (1995).
4. Carlier, J. and Chretienne, P., "Problemes d'ordonnancement: modelisation / complexite / algorithms", Masson, Paris (1988).
5. Dhamala, T. N., "Just-in-time sequencing algorithms for mixed-model production system", *The Nepali Math. Sci. report* 24, 1 (2005), 25-34.
6. Dhamala, T. N. and Khadka, S.R., "Just-in-time sequencing for mixed-model production systems revisited", submitted to *Discrete Optimization* 2007.
7. Dhamala, T. N. and Kubiak, W., "A brief survey of just-in-time sequencing for mixed-model systems", *International Journal of Operational Research* 2, 2 (2005) 38-47.
8. Graham, R.E., Lawer, E.L., Lenstra, J.K., and Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling, a survey", *Annals of Discrete Mathematics* 5 (1979) 287-326.
9. Hall, R.W., "Cyclic scheduling for improvement", *International Journal of Production Research* 26, 3 (1988) 457-472.

10. Horn, W. A., "Some simple scheduling algorithms", *Naval Research Logistics Quarterly* 21 (1974) 177-185.
11. Inman, R. R. and Bulfin, R. L., "Sequencing just-in-time mixed-model assembly lines", *Management Science* 37, 7 (1991) 901-904.
12. Jost, V., "Deux problèmes d'approximation diophantienne: le partage proportionnel en nombres entiers et les pavages équilibrés de z ", DEA ROCO, Laboratoire Leibniz-IMAG (2003).
13. Kovalyov, M. Y., Kubiak, W., Yeomans, J. S., A computational analysis of balanced JIT optimization algorithm, *Information Processing and Operational Research*, 39, 3 (2004) 4955-4974.
14. Kubiak, W., "Cyclic just-in-time sequence are optimal", *Journal of Global Optimization* 27 (2003) 333-347.
15. Kubiak, W., "Minimizing variation of production rates in just-in-time systems: A survey", *European Journal of Operational Research* 66 (1993) 259-271.
16. Kubiak, W. and Sethi, S., "Level schedules for mixed model assembly lines in just-in-time" production system", *Management Science* 37, 1 (1991) 121-122.
17. Kubiak, W., Steiner, G. and Yeomans, J.S., "Optimal level schedules for mixed-model, multi-level just-in-time assembly systems", *Annals of Operations Research* 69 (1997) 241- 259.
18. Lebacque, V., Jost, V., Brauner, N., "Simultaneous optimization of classical objectives in JIT scheduling", submitted to Elsevier Science, 2005.

19. Miltenburg, J., "Level schedules for mixed-model assembly lines in just-in-time production system", *Management Science* 35, 2 (1989) 192-207.
20. Miltenburg, J. and Goldstein, T., "Developing production schedules which balance part usage and smooth production loads for just-in-time production systems", *Naval Research Logistics* 38 (1991) 893-910.
21. Miltenburg, J. and Sinnamon, G., "Scheduling mixed-model multi-level just-in-time production systems", *International Journal of Production Research* 27, 9 (1989) 1487-1509.
22. Miltenburg, J., Steiner, G and Yeomans, S., "A dynamic programming algorithm for scheduling mixed-model just-in-time production systems", *Mathematical and Computer Modeling*, 13 (1990) 57-66.
23. Monden, Y., "Toyota production system", *Industrial Engineers and Management Press*, Norcross, GA (1983).
24. Pinedo, M., "Scheduling - theory, algorithms, and systems", *Prentice Hall*, Englewood Cliffs (1995).
25. Steiner, G. and Yeomans, S., "Level schedules for just-in-time production process", *Management Science* 39 (1993) 728-735.
26. Sukanuma, T. and Ogasawara, T., "Overview of the IBM Java Just-in-Time Compiler", *IBM System Journal*, 39, 1 (2000).
27. Tanenbaun, A., "Modern Operating System", *Prentice-Hall of India Pvt. Ltd.* (2004).

28. Thorpe, S.R., Stevenson, D.S., Edwards, G.K., “Using Just-in-Time to Enable Optical Networking for Grids”, In Workshop on Grids and Networks held in conjunction with CCGrid, April, 2004.
29. Yee, G.V., Shucker, B., Dunn, J., Sheth, A., Han, R., “Just-in-Time Sensor Networks” In Information Processing in Sensor Networks: Second International Workshop, IPSN, 2003.
30. Toyota Motor Corporation Global Site, www.toyota.co.jp.
31. www.assignmentproblem.com.
32. Wikipedia, the Free Encyclopedia(<http://en.wikipedia.org> 2008).

Appendix A

Basic Mathematical Notations

Set theory

| | |
|-------|------------------------------|
| N | Set of natural numbers |
| R | Set of real numbers |
| R^+ | Set of positive real numbers |

Sequence and series

| | |
|----------------------------|--|
| $\{a_1, a_2, \dots, a_n\}$ | Set of objects a_1, a_2, \dots, a_n |
| (a_1, a_2, \dots, a_n) | A sequence of numbers a_1, a_2, \dots, a_n |

Data

| | |
|----------------|--|
| n | Number of jobs |
| m | Number of machines |
| J_i | Job number $i, i = 1, \dots, n$ |
| n_i | Number of operations of job J_i |
| m^l | Number of machines at stage l |
| M_j | Machine number $j, j = 1, \dots, m$ |
| $O_{i,j}$ | Operation j of job J_i |
| r_i | Release time of job J_i |
| d_i | Due date of job J_i |
| s_i | Desired start time of job J_i |
| $p_{i,j}$ | Processing time of operation $O_{i,j}$ |
| W_i or w_i | Weight associated to job J_i |
| D | Total demand |

Machine environment

| | |
|---|---|
| | Single machine |
| P | Identical machines |
| Q | Uniform machines |
| R | Unrelated machines |
| m | The number of machines or stages is fixed |

Miscellaneous

| | |
|-----------|--|
| $t_{i,j}$ | Start time of operation $O_{i,j}$ |
| $C_{i,j}$ | Completion time of operation $O_{i,j}$ |
| C_i | Completion time of job J_i |
| T_i | Tardiness of job J_i |
| E_i | Earliness of job J_i |
| L_i | Lateness of job J_i |
| $E(i, j)$ | Release date of the copy (i, j) |
| $L(i, j)$ | Due date of the copy (i, j) |

Modeling the cost Value

Mathematical Function:

$$k_{i,j} = \frac{2j-1}{2r_i} = \left[\frac{\left(j - \frac{1}{2} \right) D}{d_i} \right]$$

Program:

```
public double getEvaluation(String abc){

    double eval=0.0;
    String newStr="";
    for(int i=0;i<abc.length();i++){
        newStr          =newStr+abc.charAt(i);
        char jobChar    =(char)abc.charAt(i);
        Job j           =jobModel.getJob(jobChar);
        int pos         =getLastPos(newStr,jobChar);
        double Zval     = ((double)(2*pos-1))/(2.0*j.getR());
        double Cur_eval =Math.pow((((double)(i+1))-Zval),2);
        eval            = eval + Cur_eval;

    }
    return eval;
}
```

Mapping:

getLastPos(jobChar) returns the j value for the below mathematical formula.

j.getR() returns the ratio of D/di for job j

Then our formula becomes:

$$\begin{aligned} ((j-1/2)D)/di &= \\ &= (2j-1)D/2*di \\ &= (2j-1)*(D/di)*1/2 \\ &= (2j-1)*(1/(di/D))*(1/2) \\ &= (2j-1)*(1/ri)*(1/2) \\ &= (2j-1)/(2*ri) \\ &= (2*pos-1)/2*j.getR() \end{aligned}$$

Eval is the cumulative sum calculated by the help of loop.

Appendix B

Program Source Code

1. ScheduleImage.java

```
class ScheduleImage{
private String schedule;
private int index[];
int noOfChain;
double cost;

ScheduleImage(int i){
    this.noOfChain=i;
    schedule=new String();
    index=new int[this.noOfChain];
}

public int[] getIndex() {
    return index;
}

public void setIndex(int[] index) {
    for(int i=0;i<index.length;i++)
        this.index[i]=(int)index[i];
}

public String getSchedule() {
    return schedule;
}

public void setSchedule(String schedule) {
    this.schedule = schedule;
}

public void incrIndex(int i){
    this.index[i]++;
}

public void printIndex(){
    for(int i=0;i<this.noOfChain;i++){
        System.out.print(this.index[i]);
    }
}

public double getCost() {
    return cost;
}

public void setCost(double cost) {
    this.cost = cost;
}
}
```

2. PossibleSchedule.java

```
import java.util.Vector;
public class PossibleSchedule {

    Vector <ScheduleImage>cat=new Vector<ScheduleImage>();
    int totalDemand=0;
    double Z[ ][ ] =      new double[100][100];
    double r[ ] =         new double[100];
    int pos[ ][ ] =       new int[100][100];

    private String chain[]={
        "aba",
        "cedcc"
    };

    int startIndex=0;
    int finalIndex=0;
    JobModel jobModel=new JobModel();

    public double getEvaluation(String abc){

        double eval=0.0;
        String newStr="";
        for(int i=0;i<abc.length();i++){
            newStr      =   newStr+abc.charAt(i);
            char jobChar =   (char)abc.charAt(i);
            Job j       =   jobModel.getJob(jobChar);
            int pos     =   getLastPos(newStr,jobChar);
            double Zval =   ((double)(2*pos-1))/(2.0*j.getR());
            double Cur_eval= Math.pow((((double)(i+1))-Zval),2);
            eval        =   eval + Cur_eval;
        }
        return eval;
    }

    public int getLastPos(String newStr,char c){

        int retVal=0;
        for(int i=0;i<newStr.length();i++){
            char chr=newStr.charAt(i);
            if(chr==c){
                retVal++;
            }
        }
        return retVal;
    }

    PossibleSchedule(){
        int noOfChain=chain.length;
        int length=0;
        for(int i=0;i<noOfChain;i++){
```

```

        length+=chain[i].length();
        ScheduleImage s=new ScheduleImage(noOfChain);
        int index[]=new int[noOfChain];
        index[i]=1;
        String schedule="" +chain[i].charAt(0);
        s.setIndex(index);
        s.setSchedule(schedule);
        cat.add(s);
        finalIndex++;
    }
    int jobIndex=0;
    for(int i=0;i<noOfChain;i++){

        for(int j=0;j<chain[i].length();j++){

            char c=((String)chain[i]).charAt(j);
            int index=this.jobModel.getIndex(c);
            if(index==-1){
                jobModel.addJob(c,jobIndex++);
            }
            jobModel.incJobCount(c);
        }
    }
    jobModel.setRatio();

    int prevIndex;
    for(int i=1;i<length;i++){
        prevIndex=startIndex;
        startIndex=finalIndex;
        try{
            for(int k=prevIndex;k<startIndex;k++){
                ScheduleImage s=new ScheduleImage(noOfChain);
                s=(ScheduleImage)cat.get(k);
                for(int j=0;j<noOfChain;j++){
                    int index[]=new int[noOfChain];
                    index= s.getIndex();
                    if(index[j]<chain[j].length()){
                        try{
                            String
schedule=s.getSchedule()+chain[j].charAt(index[j]);
                            ScheduleImage s1=new ScheduleImage(noOfChain);
                            s1.setIndex(s.getIndex());
                            s1.incrIndex(j);
                            s1.setSchedule(schedule);
                            cat.add(s1);
                            finalIndex++;
                        }catch(Exception e){
                            System.out.println("Error TrAp");
                        }
                    }
                }
            }
        }catch(Exception e){
            System.out.println("erer");
        }
    }

```

```

    }
    showFinalSchedule(startIndex);
}

public static void main(String abc[]){
    new PossibleSchedule();
}

public void showVector(){

for(int i=0;i<cat.size();i++){
    ScheduleImage s=(ScheduleImage)cat.get(i);

    s.printIndex();
    System.out.print(" ");
    System.out.println(s.getSchedule());
}

}

public void showFinalSchedule(int startIndex){
int count=1;
for(int i=startIndex;i<cat.size();i++){
    ScheduleImage s=(ScheduleImage)cat.get(i);
    s.setCost(getEvaluation(s.getSchedule()));

    System.out.print(count+" ");
    System.out.print(" ");
    System.out.println(s.getSchedule()+" --- "+s.getCost());
    count++;
}
}

```

3. JobModel.java

```

class JobModel{
    Vector <Job>cat=new Vector<Job>();

    public void addJob(char c,int jobIndex){

        Job j=new Job();
        j.setJobChar(c);
        j.setJobCount(0);
        j.setR(0.0);
        j.setJobIndex(jobIndex);
        this.cat.add(j);

    }

    public void setRatio(){
        for(int i=0;i<cat.size();i++){
            totalDemand+=((Job)cat.get(i)).getJobCount();
        }
        for(int i=0;i<cat.size();i++){
            Job j=(Job)cat.get(i);

```

```

        j.setR((double)j.getJobCount()/((double)totalDemand);
    }
}

public void addJob(Job j){
    cat.add(j);
}

Job getJob(char c){
    for(int i=0;i<cat.size();i++){
        Job j=cat.get(i);
        if(j.getJobChar()==c)
            return j;
    }
    return null;
}

int getIndex(char c){
    for(int i=0;i<cat.size();i++){
        Job j=cat.get(i);
        if(j.getJobChar()==c)
            return i;
    }
    return -1;
}

void incJobCount(char c){
    for(int i=0;i<cat.size();i++){
        Job j=cat.get(i);
        if(j.getJobChar()==c)
            j.incrJobCount();
    }
}

public void viewJobList(){
    System.out.println("Job      Demand      Ratio");
    for(int i=0;i<cat.size();i++){

        Job j=cat.get(i);
        System.out.println(j.jobChar+"      "+j.getJobCount()+"      "+j.getR());
    }
}
}
}
}

```

4. Job.java

```

class Job{
    double r;
    char jobChar;
    int jobCount=0;
    int actualPos=-1;
    int pos=-1;
    int jobIndex=-1;

    public int getJobCount() {
        return jobCount;
    }
}

```

```

public void setJobCount(int jobCount) {
    this.jobCount = jobCount;
}
public char getJobChar() {
    return jobChar;
}
public void setJobChar(char jobChar) {
    this.jobChar = jobChar;
}
public double getR() {
    return r;
}
public void setR(double r) {
    this.r = r;
}

public void incrJobCount(){
    this.jobCount++;
}
public int getActualPos() {
    return actualPos;
}
public void setActualPos(int actualPos) {
    this.actualPos = actualPos;
}
public int getPos() {
    return pos;
}
public void setPos(int pos) {
    this.pos = pos;
}
public int getJobIndex() {
    return jobIndex;
}
public void setJobIndex(int jobIndex) {
    this.jobIndex = jobIndex;
}
}}}

```

5. MinSumAbsoluteChainAlgo.java

```

public class minSumAbsoluteChainAlgo {
    private int demand[]=new int[100];
    private String scheduleEDD,scheduleCost;
    private String realEddSchedule,realCostSchedule;
    private String chain[]={
        "aba",
        "cedcc"
    };
    public static void main(String abc[]){
        new minSumAbsoluteChainAlgo();
    }
    minSumAbsoluteChainAlgo(){

        for(int i=0;i<chain.length;i++){
            demand[i]=chain[i].length();

```



```

}

EDD edd=new EDD(demand);
scheduleEDD=edd.getSchedule();
this.realEddSchedule=convertSchedule(scheduleEDD);
System.out.println("EDD Schedule :"+realEddSchedule);
}

private String convertSchedule(String pseudoSchedule){
String realSchedule="";
int chainPtr[]=new int[100];
for(int i=0;i<chain.length;i++){
    chainPtr[i]=0;
}
for(int i=0;i<pseudoSchedule.length();i++){
    int index=Integer.parseInt(pseudoSchedule.charAt(i)+"");
    char job=chain[index-1].charAt(chainPtr[index-1]++);
    realSchedule=realSchedule+job;
}
return realSchedule;
}
}
}

```

6. EDD.java

```

import util.DoubleUtil;
public class EDD {
private String schedule="";
public EDD(int demand[]){

int product=demand.length;
int current[]=new int[100];
double dueDate[][]=new double [100][100];
int totalDemand=0;

for(int i=0;i<product;i++){
    totalDemand+=demand[i];
}

int index=1;

for(int i=0;i<product;i++){
    for(int k=1;k<=demand[i];k++,++index ){
dueDate[i][k-1]=DoubleUtil.getRoundDouble((((double)k-
0.5)*(double)totalDemand)/demand[i],3);
    }
}

for(int i=0;i<product;i++){
    current[i]=0;
}

int minIndex;
double minValue;

```

```

for(int j=0;j<totalDemand;j++){
    minIndex=0;
    minValue=Double.MAX_VALUE;//dueDate[0][current[0]];

    for(int i=0;i<product;i++){
        if(minValue>dueDate[i][current[i]]&&current[i]<demand[i]){
            minIndex=i;
            minValue=dueDate[i][current[i]];
        }
    }
    current[minIndex]=current[minIndex]+1;
    schedule=schedule + (minIndex+1);
}

public String getSchedule() {
    return schedule;
}
public void setSchedule(String schedule) {
    this.schedule = schedule;
}
}

```

7. DoubleUtil.java

```

package util;
public class DoubleUtil {

    public static double getRoundDouble(double value,int decimal){
        double fact=0.50000/Math.pow(10.0,decimal);
        int temp=(int)((value+fact)*Math.pow(10.0,decimal));
        value=temp/Math.pow(10.0,decimal);
        return value;
    }
    public static void main(String abc[]){

        System.out.println(upperFloor(56.1));
        System.out.println(Double.MAX_VALUE);
        System.out.println(Math.E);
    }

    public static int upperFloor(double d){
        if((double)(int)d==d)
            return (int)d;
        return (int)(d+1.0);
    }
}

```

8. Earliest Due Date. java

```

import java.util.Vector;
import earliestDueDate.model.*;
import earliestDueDate.POJO.Job;
import util.DoubleUtil;
import util.ListViewModel;
import util.ShowSchedule;

```

```

public class EarliestDueDate {
    int demand[]=new int[100];
    int product;
    int totalDemand;
    double dueDate[][]=new double[100][100];
    int current[]=new int[100];
    int schedule[]=new int[100];
    Vector <Job>jobList=new Vector<Job>();
    private ListViewModel Datamodel=new ListViewModel();
    String color[]=null;

    public EarliestDueDate(){
        this.jobList=new JobDataModel().getJob();
        runEarliestDueDateAlgo();
    }
    public void runEarliestDueDateAlgo(){
        this.product=this.jobList.size();
        totalDemand=0;
        for(int i=0;i<product;i++)
            demand[i]=this.jobList.get(i).getDemand();
            totalDemand+=demand[i];
        }
        this.Datamodel.setMatrix(this.totalDemand,3);
        String []dataColName={"Product","Unit","Due Date"};
        Datamodel.setColName(dataColName);
        color=new String[this.totalDemand];
        int index=0;
        for(int i=0;i<product;i++){
            this.Datamodel.setValueAt(index,""+(i+1));
            index=index+3*demand[i];
        }
        index=1;
        for(int i=0;i<product;i++){
            for(int k=1;k<=demand[i];k++,++index ){
                dueDate[i][k-1]=DoubleUtil.getRoundDouble((((double)k-
                0.5)*(double)totalDemand)/demand[i],3);
                if(k!=1){
                    this.Datamodel.setValueAt(index-1,"-");
                }
                this.Datamodel.setValueAt(index++, ""+k);
                this.Datamodel.setValueAt(index++, ""+dueDate[i][k-1]);
            }
            try{
            }catch(Exception e){
                this.Datamodel.setValueAt(index,""+(i+2));
            }
        }
        for(int i=0;i<product;i++){
            current[i]=0;
        }
        int minIndex;
        double minValue;
        for(int j=0;j<totalDemand;j++){

```

```

        minIndex=0;
        minValue=Double.MAX_VALUE;//dueDate[0][current[0]];
        for(int i=0;i<product;i++){
            if(minValue>dueDate[i][current[i]]&&current[i]<demand[i]
            ){
                minIndex=i;
                minValue=dueDate[i][current[i]];
            }
        }
        current[minIndex]=current[minIndex]+1;
        schedule[j]=minIndex+1;
        color[j]="#" + minIndex + "" + minIndex + "" + minIndex + "" +
        minIndex + "" + minIndex + "" + minIndex;
    }
    String scheduleList="<HTML>Schedule List :<br>";
    for(int i=0;i<totalDemand;i++){
        scheduleList+=schedule[i]+" - ";
    }
    scheduleList+="</html>";

    new ShowSchedule(this.Datamodel,"Earliest Due
    Date",scheduleList);
}

```

9. Input demand. Java

```

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.JFrame;
import earliestDueDate.POJO.Job;
import earliestDueDate.com.EarliestDueDate;
import earliestDueDate.model.JobDataModel;
import earliestDueDate.model.TabelModelForJob;

public class InputDemand extends JFrame implements ActionListener{
    JTable table=new JTable();
    TabelModelForJob model=new TabelModelForJob();
    JButton jbtAdd=new JButton("Add New");
    JButton jbtSave=new JButton("Save");
}

```

```

JButton jbtSchedule=new JButton("Schedule");
JTextField jtfJob=new JTextField(5);
JTextField jtfJobDemand=new JTextField(5);

private String MODE="ADD";
private void init(){
    JPanel jpCenter=new JPanel();
    jpCenter.setLayout(new FlowLayout());
    JScrollPane jspTable=new JScrollPane(this.table);
    this.table.setModel(this.model);
    jpCenter.add(jspTable);
    JPanel jpButton =new JPanel();
    jpButton.setLayout(new FlowLayout());
    jpButton.add(this.jbtAdd);
    jpButton.add(this.jbtSave);
    jpButton.add(this.jbtSchedule);
    JPanel jpData=new JPanel();
    jpData.setLayout(new FlowLayout());
    jpData.add(new JLabel("Job Id:"));
    jpData.add(this.jtfJob);
    jpData.add(new JLabel("Demand :"));
    jpData.add(this.jtfJobDemand);
    this.jtfJob.setEditable(false);
    this.setLayout(new BorderLayout());
    JPanel jpDownHolder=new JPanel();
    jpDownHolder.setLayout(new GridLayout(2,1,5,5));
    jpDownHolder.add(jpData);
    jpDownHolder.add(jpButton);
    JPanel jpDown=new JPanel();
    jpDown.setLayout(new FlowLayout());
    jpDown.add(jpDownHolder);
    this.add(jpDown,BorderLayout.SOUTH);
    this.add(jpCenter,BorderLayout.CENTER);
    this.jbtSave.addActionListener(this);
    this.jbtAdd.addActionListener(this);
    this.jbtSchedule.addActionListener(this);
    this.setTitle("Earliest Due Date - Input Demand ");
}

public static void main(String abc[]){
    InputDemand mf=new InputDemand();
    mf.setSize(400,400);
    mf.setVisible(true);
    mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public InputDemand(){
    init();
    this.table.addMouseListener(new MouseAdapter(){
    public void mouseClicked(MouseEvent arg0) {
        InputDemand.this.MODE="EDIT";
        int i=InputDemand.this.table.getSelectedRow();
        Job j=InputDemand.this.model.getJobAt(i);
        InputDemand.this.jtfJob.setText(""+j.getJobId());
        InputDemand.this.jtfJobDemand.setText(""+j.getDemand());
    }
});
}

```

```

}
public void actionPerformed(ActionEvent ae) {
    if(ae.getSource().equals(this.jbtSchedule)){
        System.out.println(new JobDataModel().getJob().size());
        long l1=System.currentTimeMillis();
        EarliestDueDate dp=new EarliestDueDate();
        long l2=System.currentTimeMillis();
        JOptionPane.showMessageDialog(this,"Total Run
        Time :"+(l2-l1)+" milisecond");
    }
    if(ae.getSource().equals(this.jbtAdd)){
        clearBox();
        this.MODE="ADD";
    }
    if(ae.getSource().equals(this.jbtSave)){
        Job j=new Job();
        if(this.MODE.equals("ADD")){
            j.setDemand(Integer.parseInt(this.jtfJobDemand.getText()));
            this.model.addJob(j);
        }
        else{
            System.out.println("EDIT");
            try{
                j.setJobId(Integer.parseInt(this.jtfJob.getText()));
                j.setDemand(Integer.parseInt(this.jtfJobDemand.getText()));
                this.model.editJob(j);
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        this.table.updateUI();
        clearBox();
    }
    public void clearBox(){
        this.jtfJob.setText("");
        this.jtfJobDemand.setText("");
    }
}
}

```