CHAPTER 3

THEORETICAL CONSIDERATIONS AND LITERATURE SURVEY

3.1 Theoretical Considerations

Scheduling theory is much more considered as a mathematical model related to scheduling function. Several models as well as techniques are developed to be an interface between theory and practice. In this thesis, single machine with due dates is considered. Even there are many operations in the production, they can be considered as a single machine since all operations are continuously linked together with only 1 machine per each operation. This is not included the backend process which will separately consider as another process.

There are many objectives for scheduling depending on each particular business and its problems. Before moving forward, table 3 shows the key performance measurement usually used in scheduling.

Performance Measurement in Mathematical Terms		
d_i	due date	The promised delivery date
a_i	allowance	Allowed time for porcessing between ready time and the due date
W_{ik}	waiting time	The waiting time of job"I" preceding its "k" operation
W_i	total waiting time	The total waiting time
C_i	completion time	The time at which processing of J_i finishes.
F_i	flow time	The spending time of J_i in the production line.
L_i	lateness	The difference between its completion time and its due date : $L_j = C_j - d_j$
T_i	tardiness	Max $\{L_i, 0\}$
E_i	earliness	Max $\{-L_i, 0\}$

Table 3 The Performance Measurement in Mathematical Terms [Simon French, 1988]

In this case, tardiness and weighted tardiness is the key performance measurement.

Tardiness Criteria:

If a job completes later than its due date, it would say that the job is *"tardy"*. It can be defined as $T_i = \max(0, C_j - d_j)$. In the other hands, $T_i = \max(0, L_j)$

3.1.1 Dynamic Programming Approach:

Dynamic programming originated by Bellman is applicable to various optimizing problems, not just only for scheduling problem. The concept of this method is to break down any problems into a sequence of nested problem. The solution of each nested problem will be derived in a straight-forward fashion from that of the preceding problem. It could be said that this technique works "bottom up" rather than "top down".

Dynamic programming is consisted of four main methods as described below:

- Characterize the structure of the optimal solution
- Recursively define the value of optimal solution
- Compute the value of solution in a "bottom up" fashion
- Constructed the optimal solution using the computed solution

Normally, in scheduling point of view, a regular measurement of performance is a function of job completion time versus its priority.

$$Z = f(C_1, C_2, \ldots, C_n)$$

This formula interprets a measurement Z as a cost function. However, in some problems, a measurement Z can be described by

$$Z = \sum_{j=1}^{n} g_j(c_j)$$

In this case, total tardiness penalty is considered then

$$g_j(c_j) = w_j(c_j - d_j) \qquad \text{if } c_j > d_j$$

$$g_j(c_j) = 0 \qquad \text{if } c_j <= d_j$$

From this form, the dynamic programming can be applied in order to get an optimum sequence.

Define "*J*" as a subset of *n* jobs and *J'* as the complement of set *J*.

Define "q_J" as the total time required to process the jobs in set *J'*.

$$q_J = \sum_{j \varepsilon J} t_j$$

Suppose that all the jobs in set *J'* has been constructed before every job in set *J*. Then, if such a sequence is an optimal sequence, the principle of optimality of dynamic programming requires that.

No matter how the jobs in *J'* are sequenced, the jobs in *J* must be sequenced optimally, subject to the constraint that none may begin prior q_J.

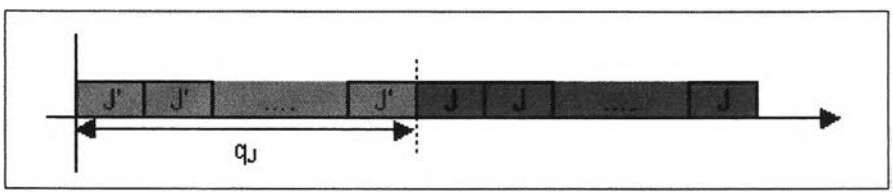

Figure 8 The structure of a job sequence of the purposes of dynamic programming [Kenneth, 1974]

Define G(J) as the minimum cost for the jobs in set *J*, subject to the constraint that non begin prior to q_J. Also define *K* as the set of all jobs. Then,

$$G(K) = \min_{j \varepsilon K} [g_j(t_j) + G(K - \{j\}] \quad \text{.....(1)}$$

or,

$$G(J) = \min_{j \varepsilon J} [g_j(q_J + t_j) + G(K - \{j\}] \quad \text{.....(2)}$$

when,

$$G(\emptyset) = 0 \quad \text{.....(3)}$$

At each stage, the function G(J) measures the total tardiness contributed by the jobs in set *J*, when set *J* is considered at the end of the schedule and is sequenced optimally. The "recursion relations" of (1) and (2) shows that in order to calculate the value of G for each particular subset of size *k*, it is necessary to know the value of G for *k* subsets of size *k-1* before hand. Thus, the procedure starts with knowledge of the value of G for a subset of size zero, from (3).

After that, by using (2), the value of G for all subsets of size 1 can be calculated, and then the value of G for all subsets of size 2, and so on. In this manner, it clearly shows the backward procedure. This method determines which job should be scheduled first and calculated from (1) the optimal value of Z as G(K).

To illustrate the method, below is an example of dynamic programming when $\overline{T}$ (total Tardiness) is considered as the performance measurement.

Job j	t_j	d_j	w_j
1	1	2	1
2	2	7	1
3	3	5	1
4	4	6	1

Table 4 The example of Dynamic Programming

Consider the set J = {1,2,4} in the stage #3. The set J' in this cast contains only job 3, thus q_J for this set is equal to 3. In this set, we have to consider all the possible casts for the first job. So, there are three subsets;

- Job 1 comes first in the set *J*.
- Job 2 comes first in the set *J*.
- Job 4 comes first in the set *J*.

In cast that job 1 come first, then $g_1(C_1) = 2$ and for the remaining jobs G({2,4}) = 4, so that the total contribution from this set, when job 1 comes first, is 6. Be noted that G({2,4}) value can be looked up from stage # 2.

In case that job 2 comes first, then $g_2(C_2) = 0$ and for the remaining jobs G({1,4}) = 8, so that the total contribution from this set, when job 2 comes first, is 8.

In case that job 4 comes first, then $g_4(C_4) = 1$ and for the remaining jobs G({1,2}) = 9, so that the total contribution from this set, when job 4 comes first, is 10.

The minimum from these three subsets is 6, which is designated as G(J) in the table, this can be achieved when job 1 comes first.

At the final stage, stage # 4, the G(J) is 4 and this indicates that job 1 should come first in the sequence. So, there are only 3 jobs, job 2,3,and 4, have to be sequenced. From set {2,3,4) in stage 3, it shows that job 3 should come first in this set, thus job 3 should occupy the second job in the optimal sequence. Continuing in this fashion, the optimal sequence can be constructed as 1-3-2-4 with the total tardiness as 4, G(K).

Stage # 1

J	{1}	{2}	{3}	{4}
q_J	9	8	7	6
$j \varepsilon J$	1	2	3	4
g_j	8	3	5	4
G(J-{j})	0	0	0	0
G(J)	8	3	5	4

Stage # 2

J	{1,2}		{1,3}		{1,4}		{2,3}		{2,4}		{3,4}	
q_J	7		6		5		5		4		3	
$j \varepsilon J$	1	2	1	3	1	4	2	3	2	4	3	4
g_j	6	2	5	4	4	3	0	3	0	2	1	1
G(J-{j})	3	8	5	8	4	8	5	3	4	3	4	5
G(J)	3	10	10	12	4	11	5	6	4	5	5	6

Stage # 3

J	{1,2,3)			{1,2,4}			{1,3,4}			{2,3,4}		
q_J	4			3			2			1		
$j \varepsilon J$	1	2	3	1	2	4	1	3	4	2	3	4
g_j	3	0	2	2	0	1	1	0	0	0	0	0
G(J-{j})	5	10	9	4	8	9	5	8	10	5	4	5
G(J)	5	10	11	4	8	10	6	8	10	5	4	5

Stage # 4

J	{1,2,3,4}			
q_J	0			
$j \varepsilon J$	1	2	3	4
g_j	8	3	5	4
G(J-{j})	0	0	0	0
G(J)	4	6	8	8

Optimum Sequence: 1 - 3 - 2 - 4

Figure 9 Dynamic Programming Calculation

It is very important to realize the computational properties of the dynamic programming. The number of subsets is 2^n.

No. of job(s)	No. of subsets	No. of job(s)	No. of subsets	No. of job(s)	No. of subsets
1	2	11	2048	21	2097152
2	4	12	4096	22	4194304
3	8	13	8192	23	8388608
4	16	14	16384	24	16777216
5	32	15	32768	25	33554432
6	64	16	65536	26	67108864
7	128	17	131072	27	134217728
8	256	18	262144	28	268435456
9	512	19	524288	29	536870912
10	1024	20	1048576	30	1073741824

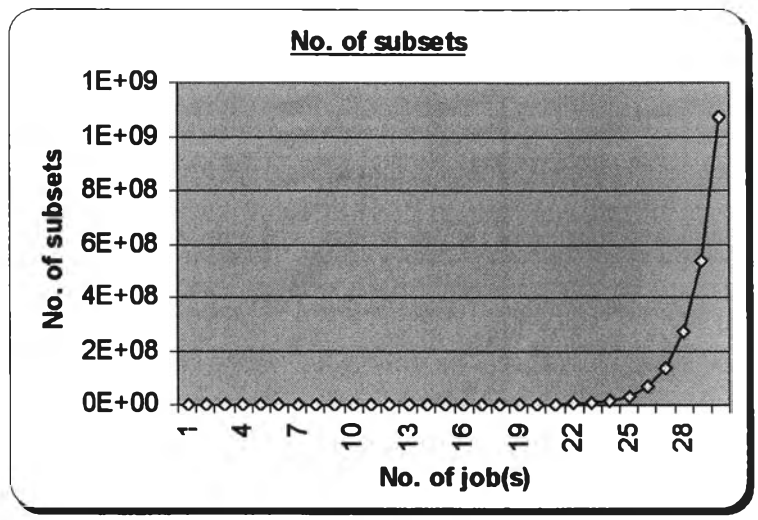

Figure 10 The computational properties of the dynamic programming procedure.

In this manner, dynamic programming is typical of many general purposed procedures for combinatorial optimization, in that the effort required to solve the problem grows at an exponential rate with increasing problem size as shown in figure 9.

This is the major constrain that makes dynamic programming becomes an inefficient method in some of the simple problems that have large sample size. However, for problems which efficient optimizing procedures have not been developed such as weighted mean tardiness or weighted number of tardy jobs, dynamic programming may be a reasonable approach.

Comparing with complete enumeration method, dynamic programming is considered to be more efficient because it will considers certain sequences only indirectly,

without actually evaluation them explicitly (all feasible sequences). This technique is often described as an *"implicit enumeration technique"*.

3.1.2 Forward and Backward Scheduling

The forward and backward scheduling are mostly used for short-term scheduling. These methods try to:

- Minimize the completion time
- Maximize the utilization
- Minimize WIP (Work In Process)
- Minimize customer waiting time

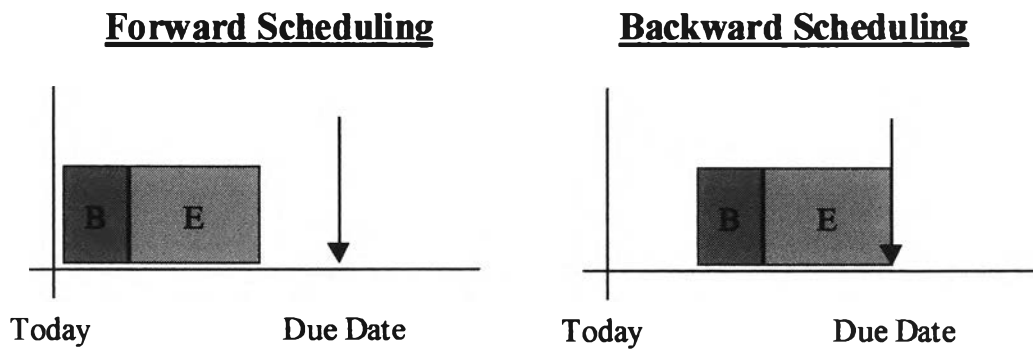

Figure 11 The forward and backward scheduling

Forward Scheduling:

The forward scheduling will begin the schedule as soon as the requirements are known. That means,

- It is a make to order job
- Schedule can be accomplished even if due date is missed
- It mostly build up the WIP (Work In Process)

Backward Scheduling:

The backward scheduling will begins by using the due date of the final operation. That means, the jobs is scheduled in reverse order. This method is used in many manufacturing considering on due date.

3.1.3 Johnson's Rule – Scheduling N Jobs on Two Machines

This method is for scheduling N jobs on two machines in the same order. The objective of this method is to minimize the make span in the shop floor. During the problem formulation, job *j* is characterized by processing time

- t_{j1}, required on machine 1
- t_{j2}, required on machine 2 after complete on machine 1

The optimal sequence can be achieved by the below rule for ordering pairs of jobs

$$\text{Min}\{t_{j1}, t_{j2}\} \leq \text{Min}\{t_{j2}, t_{j1}\}$$

In other words, the result is directly constructed with an adaptation of the sequence. The order in sequence is characterized by a one-pass mechanism that identifies a job that should be filled either at the first or last. The Johnson's rule is as below

Johnson's Rule:

Step 1: Find $\min_i\{t_{j1}, t_{j2}\}$

Step 2a: If the minimum processing time requires machine 1, place that job in the first available position in sequence. Go to step 3.

Step 2b: If the minimum processing time requires machine 2, place that job in the last available position in sequence. Go to step 3.

Step 3: Remove that job from the list and return to step 1 until all the jobs are filled.

Below is an example of Johnson's rule with 5 jobs.

Job j	1	2	3	4	5
t_{j1}	3	5	1	6	7
t_{j2}	6	2	2	6	5

Stage	Unscheduled Jobs	Minimum t_{ik}	Assignment	Partial Schedule
1	1, 2, 3, 4, 5	t_{31}	3 = [1]	3 x x x x
2	1, 2, 4, 5	t_{22}	2 = [5]	3 x x x 2
3	1, 4, 5	t_{11}	1 = [2]	3 1 x x 2
4	4, 5	t_{52}	5 = [4]	3 1 x 5 2
5	4	$t_{41} = t_{42}$	4 = [3]	3 1 4 5 2

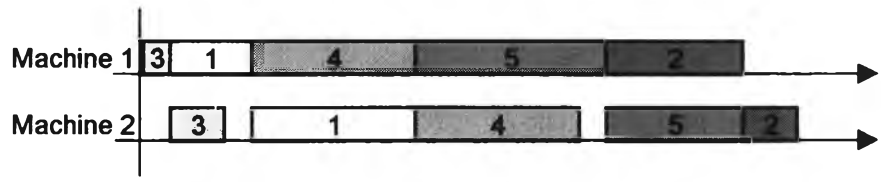

Figure 12 The Johnson's Rule example.

3.2 Literature Survey

Toshiba Corporation. [1]:

Toshiba Corporation has developed scheduling system for Toshiba's gas insulated switchgears. This system is expected to solve the significant machine setup times, strict local buffer capacities, the option of choosing a few alternative processing routes, and long horizontal. It was developed from Lagrangian relaxation method and dynamic programming. The final result shows that this system contributes to high quality schedules in a timely fashion. It makes the company to achieve on time delivery and low inventory. This system also can generate near optimal solutions with quantifiable quality in a computationally efficient manner.

University of Connecticut. Storrs. CT [2]:

Their study demonstrates the scheduling solution by using novel neural network optimization techniques with the Lagragian Relaxation method, so called Lagrangian relaxation neural network (LRNN). For unconstrained optimization, the neural networks have been based on "Lyapunov stability theory" of dynamic system. The concept is that if a network is "stable", its 'energy" will be minimize and becomes

equilibrium. For constrained optimization, Hopfield-tpe networks, known as penalty networks, will approximate a constrained problem as an unconstrained problem, so that, it can be solved by neural networks. The convergence proof of LRMM shows an effective framework for job shop scheduling contains the novel neural dynamic programming . It can overcome the difficulties associated with local minima and solution infeasibility encountered by conventional Hopfield type networks.

Real-Time Factory Floor Scheduling Enhances Responsiveness [3]:

In the past, most of scheduling systems are built around MRP II which not allow the factory to respond optimally to rapidly changing demands. The traditional concept of "not scheduling too often" does not properly work anymore since the customer responsiveness is a large parts of what differentiates one supplier from others. The concept of " virtual work order" which can be revised any time, is needed.

The real time scheduling shows the current highest manufacturing priority and how to work towards satisfying it. The system can reduce the instabilities and oscillations in the batch planning and scheduling by using the real-time feedback from actual production data. Unexpected changes such as machine downtime, materials shortages, etc can be detected easily, after that the system will re scheduling immediately. The main benefit of this system is controllability in the shop floor. The threaded scheduling, one kind of real time scheduling, add extra benefit by being an information-intensive automated process. With threaded scheduling, resource commitments are identified and made individually for each order, but their physical identity isn't fixed until an operation actually starts.

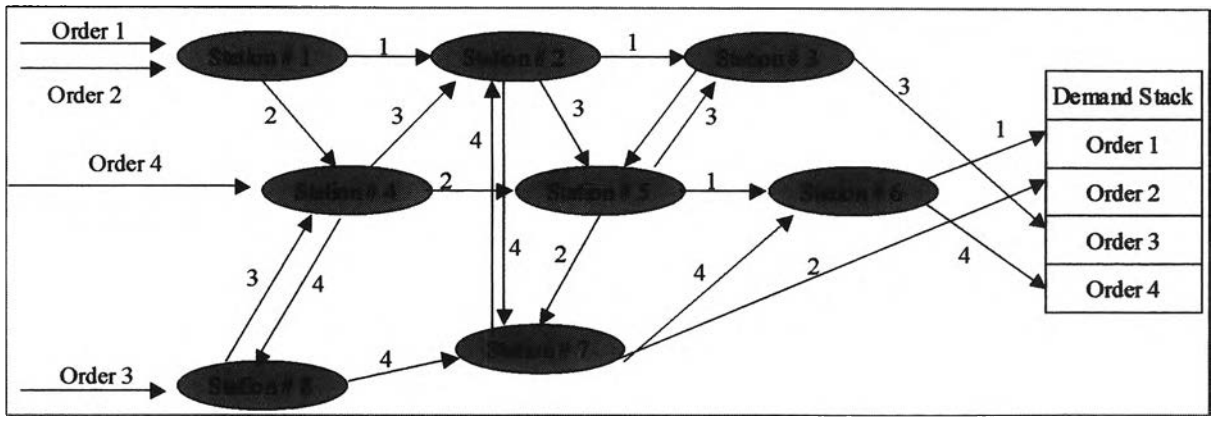

Figure 13 Threaded Scheduling Of Shared Resource

A Framework for Service Employee Scheduling [4]:

Rather than shop floor control, scheduling can be implemented in the service employee scheduling. In service operations, inadequate capacity leads to long waiting time and potential loss sales. So that, employee scheduling has to support acceptable service levels and specify results in hours and minutes. This scheduling system has to find out the lowest cost set of feasible schedules that can satisfy the requirements for each period.

On-Line Simulation For Real-Time Scheduling Of Manufacturing Systems [5]:

If manufacturing environment has no uncertainty, production planning and scheduling could be generated off-line. Unfortunately, there are many uncertainties in the operation line such as operator absenteeism, material shortages, order changes, yield variation and so on.

In current situation which has many uncertainties as described above, it might be advantageous to change the way that operation is controlled in to the optimum way. On line simulation can be used to support real-time scheduling decisions by giving the best option to deal with unexpected events on the shop floor. Thus, on-line simulation based scheduling system can operate as a "What now" tool, capable to advise about current decision.

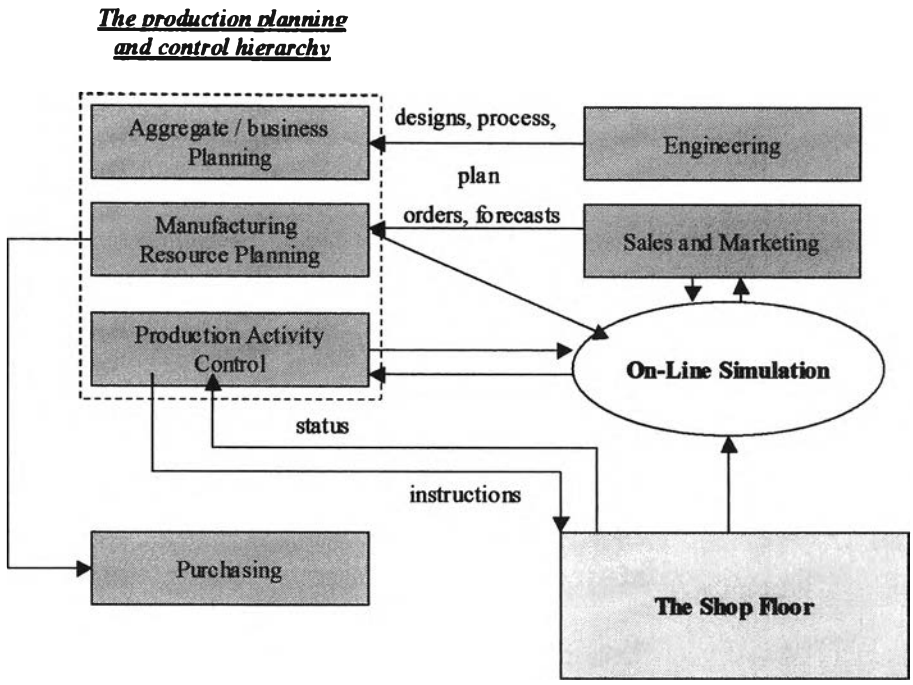

Figure 14 On-Line Simulation as a "What Now" tool.

Dynamic Production Scheduling For A Process Industry [11]:

Renato and Moique studied the complex decisions and cost tradeoffs in capacity-oriented production scheduling (CPS) by using tile manufacturing. The contribution of this study lies in developing an efficient procedure for scheduling the production of product families on several flexible lines over a given number of periods, to minimize the total production, inventory, and changeover cost. The solution approaches that have been used include dynamic programming, Lagrangian techniques, and the cutting plane method to strengthen linear programming. The developed model tries satisfying all the demands of underproduced products utilizing lines that are idle or have been assigned to overproduced product. A couple of features that distinguish the heuristic from other forward scheduling approaches are its lookahead scheme in prioritizing the underproduced products, and the efficient method of assigning and reassigning products to production line. The result from this model show about 53% of the cases had gaps less than 5% and 30% of the cases had gaps between 5 and 8%.

New Lower And Upper Bounds For Scheduling Around A Small Common Due Date [12]:

They consider the single machine problem of scheduling n jobs to minimize the sum of the deviations of the job completion times from a given small common due date. They developed a branch-and-bound algorithm based on Lagrangian lower and upper bounds that are found in O(n log n) time. The common due date is either specified as part of the problem instance, or is a decision variable that has to be optimized simultaneously with the job sequence. They developed an approximation algorithm for the common due date problem based upon Johnson's approximation algorithm for subset-sum, which runs in $O(m)$time after sorting, and worst-case behavior by using the Even-Odd Heuristic.

Scheduling Unit Time Open Shops With Deadlines [13]:

This study considers n jobs and m machines. A feasible combination of the machine and job orders is called a schedule. They developed a minimal total completion time and a minimal number of tardy jobs and the minimal value of maximum lateness by using *latin rectangle* LR[n, m, k]. This is to ensure that each machine cannot process tow jobs at the same time and each job will occur at most once in every column. The minimizing number of late job problem is solved based on the fact that if there is a schedule with exactly k tardy jogs, then there exists a schedule in which the jobs with the k smallest due dates are late and all other jobs are on time. In this study, they show that there is a common optimal schedule for both C_{max} and $\sum C_i$, even if we minimize these functions with respect to deadline.

Scheduling Groups Of Jobs On A Single Machine [14]:

Since economies of scale are fundamental to manufacturing operations, it manifests itself in efficiencies gained from grouping similar jobs together. This paper demonstrates the single machine scheduling models that incorporate benefits from job grouping. There are three main areas focused, *family scheduling with item availability*, *family scheduling with batch availability*, and *batch processing*. The grouping of jobs is a desirable or necessary tactic because of some technological feature of the processing capability such as changeover time, or setup time.

Item Availability Family Scheduling Models:

In general, any family scheduling model can be viewed as a single machine model with sequence dependent setup times. For the job following a member of the same family, its setup time is zero. However, by exploiting the special structure of family scheduling, we can sometimes avoid the enumerative techniques that would ordinarily be required. The techniques for this group are minimizing total weighted flow time under the GT assumption and minimizing maximum lateness.

Batch Availability Family Scheduling Models:

Batch availability is a bit more difficult than item availability. Typically a batch processing model is related to the weight, size, or the number of jobs in a batch. Batch availability is characteristic of systems in which jobs transported and ultimately delivered in container such as boxed, pallets, or carts. The techniques used in this model are minimizing total weighted flow time under the GT assumption and minimizing maximum lateness.

Batch Processing Models:

A batch processor can accommodate several jobs simultaneously. Its limitation is the maximum number of jobs that can be processed at any one time. Batch processing models tend to be much more complex when the capacity consumption of each job is allowed to vary. The techniques normally used for this group are minimizing makespan with dynamic arrival, minimizing total flow time with dynamic arrivals, minimizing maximum lateness with dynamic arrivals, and batch dependent processing times.

Optimal Scheduling Of Fallible Inspections [15]:

This paper demonstrates the optimal solution to the problem of designing inspection schedules with fallible and time-consuming test procedures. The inspection scheduling is important to detect promptly the occurrence of events that are not immediately manifest. It is designed to achieve a balance between the cost of inspections and the cost of undetected failure. The solutions is derived in continuous time, with arbitrary failure distribution, and is depended on infinite-horizon dynamic programming with time-dependent utilities, and with an additional optimization with respect to initial conditions.

Scheduling Jobs On Several Machines With The Job Splitting Property [16]:

This study is conducted in a textile industry in which jobs may be independently split over several specified machines, and preemption is allowed. Deadlines are also concerned for each job. Minimizing maximum weighted tardiness used in this study can be done in polynomial time. In general case of unrelated machines, this problem can be solved by linear programming or by generalized network flow techniques. However, in the case of uniform machines, a network flow model can be developed with algorithms based on max flow computations. In order to schedule all jobs as early as possible, to minimizing the maximum weighted tardiness, they address the problem of finding so-called Unordered Lexico optimal solution.

Mean Flow Time Minimization In Reentrant Job Shops With A Hub [17]:

They study the problem of scheduling a reentrant job shop which can be found in many production systems such as in VLSI (Very Large Scale Integrated circuit), wafer fabrication process, PCBs (Printed Circuit Board) and so on.

In their study, there are two step approach to scheduling primary operations on the hub machine. The first step is to optimize sequences of jobs under the *Hereditary Order* (HO) assumption. The second step is to find an optimal sequence of primary operations for a given sequence of jobs by using dynamic programming algorithm. This study shows that the SPT (Shortest Processing Time) job order is optimal for the single machine reentrant, and a dynamic programming is proposed to derive an optimal no-passing schedule for a given job order.

A Dynamic Subgradient-Based Branch-And-Bound Procedure For Set Covering [18]:

They developed a branch and bound algorithm for set covering. The new procedure is known as dynamic subgradient optimization (DYNSGRAD). This DYNSGRAD combines the standard subgradient method with primal and dual heuristics. The main advantage of subgradient optimization over the simplex method is its low computational cost due to the number of iterations required for convergence does not depend on problem size. It performs significantly better than other procedures in terms of the quality of solution obtainable with a certain computational effort.

Dynamic Programming Strategies For The Traveling Salesman Problem With Time Window And Precedence Constraints [19]:

They study on traveling sale man problem with time window and precedence constraints (TSP-TWPC). They describe an exact algorithm to solve the problem based on dynamic programming and make use of bounding function to reduce the state space graph. These functions are obtained by means of a new technique that is a generalization of the "State Space Relaxation" for dynamic programming. The main contribution in this paper is a new method to derive from the state space associated with the dynamic programming recursion of the problem. A new bounding procedure is developed by using forward and backward dynamic programming recursion in order to derive a better reduced state space by eliminating those states that cannot lead to the optimal solution.

Scheduling Semiconductor Burn-In Operations To Minimize Total Flow Time [20]:

They address a batching scheduling problem in the burn-in stage of semiconductor manufacturing. This problem involves assigning jobs to batches and determining the batch sequence in order to minimize the total flow time. They develop a dynamic programming based heuristic for the general problem which guarantees a solution that is at most twice the value of the optimal solution.

Due-Date Scheduling: Asymptotic Optimality Of Generalized Longest Queue And Generalized Largest Delay Rules [21]:

Jan use Gcu rules to optimize nonlinear criteria specified in term of delays and /or queue. This technique provides a simple but effective tool. Its objective is to simplify a generalized longest queue (GLQ) or generalized largest delay (GLD) rule.

Melt Scheduling to Trade Off Material Waste and Shipping Performance [22]:

They develop an efficient melt-scheduling heuristic for the steel manufacturing process by using MIP (Mixed-Integer-Programming) formulation. They used two level processes in the scheduling heuristic as below.

- Level 1: Ingot Selection. This is to solve knapsack problem.
- Level 2: Detailed Resource Allocation: To determine in case that the particular set of ingots selected in level 1 can be processed in the facility.

The objective function is a convex combination of *total waste* and *total tardiness*. This is to bring out the interesting trade-off between the business objective (shipping performance) and the manufacturing objective (waste reduction).