Hybridized arrival time control approach to JIT job-shop scheduling

Nazrul I. Shaikh
Industrial Engineering Dept.
Penn State University
University Park, PA 16802

Vittaldas V. Prabhu
Industrial Engineering Dept.
Penn State University
University Park, PA 16802

Patrick Reed
Civil Engineering Dept.
Penn State University
University Park, PA 16802

Abstract

There is a need for highly flexible, low complexity systems with that can handle the large scale real time scheduling problems. In this work, two approaches, artificial intelligence (AI) based genetic algorithms (GA) and control theoretic based arrival time control (ATC) have been combined to achieve the objective of just in time (JIT) scheduling in a job-shop. GA and ATC compliment each other and the hybrid system, relying on its massively parallel architecture can handle large-scale JIT scheduling problems efficiently. The required number of function evaluations decrease drastically and scale polynomially as the problem size increases. Initial results show a 21% improvement in soultion quality over simple ATC and 0.5% improvement over heurisitcal solutions for a sample problem considered here.

1. INTRODUCTION

Scheduling for just in time (JIT) delivery is fast becoming the de facto requirement in many small to medium scale manufacturing shop floors. Over the last few years, researchers and practitioners have approached the problems that arise in job shop scheduling when the objectives are earliness-tardiness related. Combinatorial optimization techniques (e.g., Ventura, 1995, Pinedo, 1995, Baker, 1974), control theoretic approaches (e.g., Prabhu, 2003, Kogan and Khmelnitsky, 2001), and artificial intelligence (AI) based approaches (Scholl et al, 1995) have been proposed. The implementations of these scheduling approaches, however, are not that widespread in industry.

Researchers focus on complexity analysis, rigorous analysis of exact procedures, or heuristics for mathematically tractable special cases of the real problems. Most large-scale, real-time scheduling problems do not fall into these special categories. Large and complex search spaces for solutions, dynamic shop floor conditions, and a multitude of shop floor conditions are the factors that set these problems apart. There is a need for highly flexible systems with low complexity that can handle such problems. It is also essential that the system be robust enough to handle variations in shop floor conditions and scalable to handle real size problems. The focus of this work is on combining features of evolutionary strategies and control theoretic approaches and developing a hybrid approach for JIT scheduling. The goal is to develop a scalable, robust, and low complexity approach that generates efficient schedules.

The proposed hybrid approach relies on problem partitioning and local search for developing schedules. Arrival time control (ATC) formulation (Prabhu, 2003) transforms the scheduling problem into a continuous variable control problem and eliminates combinatorial complexity. Though ATC has an exponential convergence rate towards local attractors, it cannot guarantee optimality for most scheduling problems. In this work, ATC has been used to partition the problem and generate initial populations near local attractors. Genetic algorithms (GAs) have been applied within these partitions on the populations of solutions to seek superior solutions adaptively. Offline analysis is conducted and the best solution is archived for generating the final schedule.

This paper has been organized as follows: The job-shop scheduling problem with earliness-tardiness penalties has been described in Section 2. The complexity of the problem using traditional approaches, simple GA, simple ATC and the hybrid approach has been discussed. ATC algorithm has been described in Section 3 and GA in Section 4. The proposed hybrid algorithm is presented in Section 5. Results are presented in Section 6 and future research in Section 7.

2. SCHEDULING AND COMPLEXITY

Within the domain of production control, scheduling refers to the specific activity of timetabling the operations dictated by the process plans so as to achieve desired levels of performance at the shop floor level (Rodammer and White, 1988). The job-shop scheduling problem is that of scheduling a set of n jobs that have to be processed on m machines, where the processing of each job consists of m operations performed on these machines in a specified sequence. The operation of job i has to be performed on machine j with deterministic processing time t_{ij}. A machine can process only one job at a time, and an operation cannot be preempted. The objectives can be one or more, including minimization of makespan, tardiness, or maximize machine utilization, customer satisfaction, etc. The focus of this work is on earliness-

tardiness related objectives in JIT systems, i.e., minimize D, the mean squared deviation (Equation (1)), subject to Equations (2), (3), (4), and (5).

$$Min \quad D = \sum_{i=1} \sum_{j=1} (d_{ij} - c_{ij})^2 \tag{1}$$

Subject to:

$$\sum_{k=1}^{TC} X_{ijk} = 1 \qquad \forall \text{ i, j} \tag{2}$$

$$\sum_{k=1}^{TC} kX_{ijk} + t_{i(j+1)} \le \sum_{k=1}^{TC} kX_{i(j+1)k} \quad \forall \text{ i and j} = 1,..., \text{J - 1} \tag{3}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{J} X_{ijk} Y_{ijm} + \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k'=k+1}^{\min(TC, k+t^{\max}-1)} X_{ijk'} Y_{ijm} \xi_{ij(k'-k)} \le 1 \tag{4}$$

$\forall$ m, k

$$\xi_{ijk} M \ge t_{ij} - k \qquad \forall \text{ i, j , and k} \tag{5}$$

Here, d_{ij} is the due date for the j^{th} step of the i^{th} part, c_{ij} is the completion time of the j^{th} step of the i^{th} part, X_{ijk} indicates whether the j^{th} step of the i^{th} part is completed in the k^{th} time step, Y_{ijm} indicates whether the j^{th} step of the i^{th} part requires machining by the m^{th} machine, TC is the total time horizon, t_{ij} refers to the processing time the j^{th} step of the i^{th} part. Equation (2) ensures that all the operations for a part are completed, Equation (3) ensures that operations are performed in the right order, and Equations (4) and (5) ensure that the machining constraints are not violated.

It has been shown that even the simplest models that deal with earliness tardiness penalties are NP-Complete (Cheng and Gupta, 1989). The number of possible schedules exceeds $O((n!)^m)$. Most of the solution techniques for JIT scheduling problems rely on relaxation techniques and heuristic methods for this very reason. The complexity estimate for simple GA following the parameterization guidelines of Reed et. al, 2000 is of the order $O((n \times m^2)^2)$. The function evaluations occurring in the ATC algorithm during an iteration is of the order $O(n \times m)$. Therefore for I iterations, the function evaluations are of the order $O(n \times m \times I)$, though without any guarantee for optimality. The hybrid approach proposed here uses ATC to partition the problem and multi-population GAs to seek an optima in each partition. A conservative approximation for the required number of function evaluations for the proposed approach is of an order of $O(K(n \times m \times I) + (K \times m) \times (n \times m)^2)$. The first term represents the function evaluations made for the ATC algorithm for partitioning step and the second term involves the function evaluations made by m GA sub-populations. K is the number of iterations used for the hybrid approach. It may be noted that as the problem size i.e., $(n \times m)$ increases, the hybrid approach with $O(K(n \times m \times I) + (K \times m) \times (n \times m)^2) << O((n \times m^2)^2)$ and $O((n \times m^2)^2) << O((n!)^m)$ substantially reduces the problem complexity.

3. ATC IN SCHEDULING

ATC is a simulation-based algorithm that transforms the scheduling problem into a continuous variable control problem. As the name suggests, the algorithm calculates the schedule for all the parts by manipulating their arrival times. Given the expected processing time and the due date for each part, the algorithm tries to minimize the deviation of completion time from the due date by adjusting the part arrival time in the next simulation run. Essentially, for a *n* part *m* machine problem, there will be *m* resulting systems (1 for each machine), and each comprising of n-dimensional multivariable control systems, where due date, completion time, due date deviation, and arrival times are n-dimensional vectors acting as the command, output, error and manipulated vectors respectively (Prabhu, 2003). The logic is graphically illustrated in Figure 1.

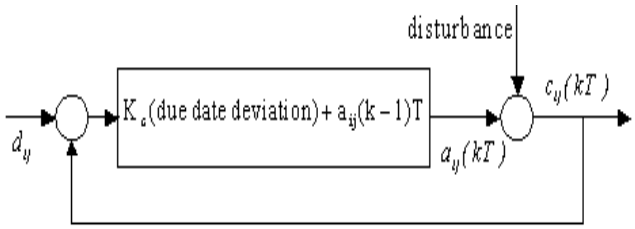

Figure 1. Feedback Control for ATC

The algorithm calculates the completion time of each part (which is nothing but the sum of arrival time of the part and the processing times of each of the parts that arrived before it and the queuing time). Based on the total time for the completion of all the parts, deviation from the actual due-date is calculated and the arrival times for the parts at next iteration are changed according to the deviations. Equation (6) represents the mathematical expression for the ATC controller.

$$a_{ij}(t) = K_c \int (d_{ij} - c_{ij}(\tau)) d\tau + a_{ij}(0) \tag{6}$$

Where, a_{ij} is the arrival time, K_c is the control system gain, On further simplification the above equation transforms into Equation (7) where T is the discrete time step. The schedule with the best performance discovered in these iterations is saved and used to schedule events.

$$a_{ij}(kT) = K_c (d_{ij} - c(k-1)T) + a_{ij}((k-1)T) \tag{7}$$

It may be noted that queuing introduces discontinuity in the system. Parts are processed in the order in which they arrive and if a part arrives when another part is being processed at the machine, it incurs queuing time. If a part arrives when the machine is idle, this queuing time is zero. All the parts that are processed by a machine between two consecutive machine idle times form a fragmented queue and are part of an "active sub-

problem". The simulation identifies these fragmented queues as the arrival times for the parts in the active sub-problem converge to steady-state values. The relative order of processing within each queue fragment however continues to change as ATC keeps searching for better schedules based on the processing sequence change. This phenomenon known as chattering (illustrated in Figure 2) occurs as ATC is oblivious to optimal.

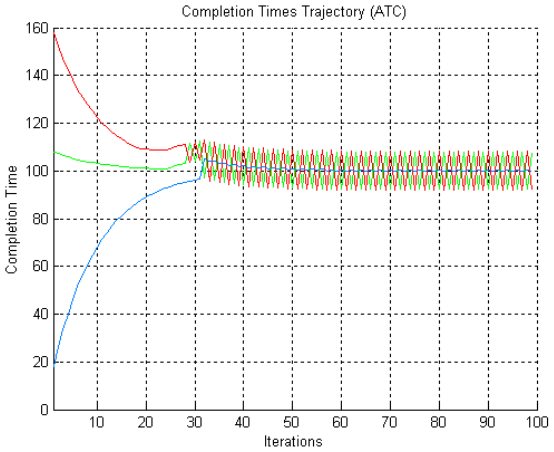

Figure 2. Chattering in ATC

After the convergence of the arrival times and formation of active sub-problems, the n part m machine job-shop scheduling problem can be seen as α_i $\ell_{\alpha i}$ part 1 machine sequencing problems, where there are $\ell_{\alpha i}$ parts in the α_i^{th} sub-problem. Recent work has focused on developing techniques to design systems to improve scheduling performance and predictability (Cho and Prabhu, 2002). Attempts are being made to incorporate a global system view with the ATC algorithm (Hong, et al, 2003).This work relies on GA for introducing a global perspective into ATC based scheduling.

4. GA FOR JOB-SHOP SCHEDULING

The GA is a heuristic search procedure for solving complex combinatorial problems. Inspired by biological evolution, GAs propagates new strings from parent chromosomes via stochastic operators. Chromosomes with high fitness values survive and those with low fitness values die off generation by generation (Holland, 1975; Goldberg, 1989). While most stochastic search methods operate on a single solution to the problem at hand, the GA operates on a population of solutions; it can efficiently sequence the operations that have been included in the population. Some of the aspects that make the GA a favorable tool for combining with ATC are:

1. Flexibility: GAs can effectively handle problems that many traditional optimization algorithms cannot include: (1) discrete spaces, (2) non-linear, discontinuous evaluation functions, and (3) nonlinear discontinuous constraints. GA has the ability to extract an initial population and display rapid convergence even in discontinuous regions.

2. Scalability: The favorable scaling of evolutionary algorithms as a function of the dimension of the search space makes them particularly effective in comparison with other search techniques for large search spaces.

3. Robustness: GAs use a population rather that a single point in the search space for evolution. This incorporates robustness into the search procedure as well as fault tolerance when inter-process communication is used.

Besides the fact that GA is efficient in the discontinuous region, the approach blends well with the distributed architecture that ATC maintains. Local populations can be generated for each individual zone of discontinuity. The details have been presented in Section 5.

To use GA, however, one must first represent the problem in the right structure. It may be noted that ATC is a simulation-based approach and the machine capacity constraints (Equations (4) and (5)) are implicitly defined in the simulations. Equations (4) and (5) are satisfied in the partitions that are created by ATC. Constraints (2) and (3) however, have to be included in the representation that is selected. The procedure then applies selection and variational operators to these individuals in the population to generate new individuals. The GAs use various selection criteria so that it picks the best individuals for mating so as to produce superior solutions by combining parts of parent solutions intelligently. The objective function of the problem being solved determines show good each individual is. In this work, we use the following specifications for GA.

4.1 REPRESENTATION

Several chromosome-encoding schemes have been discussed in literature for sequencing and scheduling of operations. These include operation-based, job-based, preference list based, machine based, or random keys based representations (Pongcharoen et al, 2003). These representations tend to become complicated as the problem size and the complexity increases. In this work, a variation of the alphanumeric representation has been used.

In the proposed hybrid approach, the partitioning stage generates sub-problems that involve sequencing of $\ell_{\alpha i}$ parts on a single machine. Since each sub-problem is specific to that machine, ordinal values of the entity types in a sub-problem are mapped into each chromosome of the sub-population. Essentially, a sub-population α_i will be comprised of chromosomes of length $\ell_{\alpha i}$ having unique integer numbers from 1 to $\ell_{\alpha i}$. For example, if there are 9 parts in an active schedule for a machine, a chromosome for that sub-population can be represented as: 1 2 3 4 5 6 7 8 9, where "1" is the ordinal value of an entity in that sub-problem.

4.2 INITIALIZATION

The chromosomes in the initial sub-populations are generated randomly. Heuristic rules such as SPT (shortest processing time first) or FIFO (first in first out) could be applied for initial population generation, but these heuristics perform well only in the common due date problems and so cannot be generalized for a sub-population. The initialization process for sub-problem α_i has two stages:

1. Strings of length $\ell_{\alpha i}$, having integer numbers sequences from 1 to $\ell_{\alpha i}$ are generated. Essentially, if *N* strings have to be created for a sub-population with 9 entities, *N* strings of characters 1 2 3 4 5 6 7 8 9 will be generated

2. The numbers in each chromosome string are then mixed randomly within the string itself, and new sequences containing all the numbers in a random order are generated. These are then used as inputs as new members of the population. For example, 2 8 4 7 1 3 9 6 5 is a new chromosome.

4.3 SELECTION

A modified "*N* best" reproduction scheme (Cheng, et al, 1996) has been used for generating the children population and determining the population for the next generation. In this scheme, η off springs are produced from the previous population of size *N*. The *N* best chromosomes out of the $N+\eta$ old ones constitute the population for the next generation. A fixed number of randomly generated chromosomes λ are also injected into the population after every κ generation to add diversity and search pressure (Reed, 2000, 2002). Therefore, $N \rightarrow N+\lambda$ after every κ generations.

4.4 DIVERSITY OPERATORS

4.4.1 Crossover

Each chromosome in has length $\ell_{\alpha i}$ and comprises of $\ell_{\alpha i}$ unique integer numbers ranging from 1 to $\ell_{\alpha i}$. This property should be maintained in the child population that is generated after the crossover and mutation operations. The "uniform order based crossover" is considered to be a good fit to this kind of constraints (Lee and Choi, 1995). This crossover operator considers both the absolute and relative positions of genes in the parent chromosomes for generating the children. The crossover operation is conducted in four steps:

1. Pick two parent chromosomes. The parents can be picked based on their fitness function values, using the roulette wheel selection, or randomly. In this work, they are randomly picked.

2. Generate a binary string of length $\ell_{\alpha i}$, the length of the chromosomes in the population (9 in this case). A binary template could be as follows: 1 1 0 0 0 0 1 0 1

3. Fill in the positions in Child 1 (or Child 2) by copying them from Parent 1 (or Parent 2) whenever the bit strings contain a 1 (or 0). For example, suppose the following parents are selected for crossover:

Parent 1: 1 2 3 4 5 6 7 8 9 and Parent 2: 2 3 5 4 1 8 9 7 6

And the Binary template is 1 1 0 0 0 0 1 0 1, then the resultant intermediate children chromosomes are:

Child 1: 1 2 _ _ _ _ 7 _ 9 and Child 2: _ _ 5 4 1 8 _ 7 _ respectively

4. List the genes from parent 1 (or parent 2) associated with a 0 (or 1) in the binary template and permute these genes so that they appear in the same order as they appear in on parent 2 (or parent 1). Thus, the crossover yields the following final children:

Child 1: 1 2 3 5 4 8 7 6 9 and Child 2: 2 3 5 4 1 8 6 7 9

4.4.2 Mutation

Intra-string mutation has been used for introducing variability in the system. The mutation operator chooses two jobs in a chromosome at random and exchanges their positions. For example, if the parent is: Parent 1: 1 2 3 4 5 6 7 8 9 and the probability of mutation is 0.1, the child can be: Child 1: 1 9 3 4 5 6 7 8 2

4.4.3 Time Continuity

Time continuity has been proposed as an added tool for introducing diversity and variability (Goldberg, 2002). The population injection method, described in Section 4.3 is a form of time continuitation that adds diversity.

4.5 SWAPPING

Before the objective function value associated with a chromosome can be evaluated, it is required that the local feasible of the sequence be tested. Although a $\ell_{\alpha i}$ part 1 machine problem is being solved in each active sub-problem, for re-entrant lines, etc, there is possibility that there are some precedence constraints that need to be adhered to. The swapping operator swaps the two genes if it is violated. For example if in parent 1, the ordinal values 2 and 9 represent the same part, but different processing steps, and the operation indicated by ordinal value 2 has to be performed before the operation indicated by 9, and the position of the two genes are interchanged in the final chromosome.

Child initial: 1 9 3 4 5 6 7 8 2

Child final: 1 2 3 4 5 6 7 8 9

It may be noted that the final child and the parent chromosome represent the same sequence. This is quite common and can lead to pre-convergence. Steps such as population injection are therefore recommended.

4.6 FITNESS EVALUATION

Each chromosome in a sub-population represents an active schedule for the machine for a given time frame. The arrival time of the first part is determined from the ATC partition and since the schedule is active, there is no inserted idle time.

The j^{th} part in a given sequence incurs a waiting time of w_{ij} after that the machine processes it for a duration of t_{ij} and then completes it at c_{ij}. The completion time for the j^{th} part in the sequence can therefore be expressed as

$$c_{ij}(t) = a_{i1}(t) + w_{ij}(t) + t_{ij}(t) \quad (8)$$

and because of zero inserted idle time,

$$w_{ij}(t) = a_{i1}(t) + t_{i1}(t) + t_{i2}(t) + \ldots + t_{i(j-1)}(t) \quad (9)$$

From Equations (8) and (9), completion time can be expressed as

$$c_{ij}(t) = a_{i1}(t) + t_{i1}(t) + t_{i2}(t) + \ldots + t_{ij}(t) \quad (10)$$

The fitness criteria, the due date deviation is then calculated as

$$D = \sum_{j=1}^{\ell_{\alpha i}} (d_{ij}(t) - c_{ij}(t))^2 \quad (11)$$

5. THE HYBRID APPROACH

The proposed approach can be represented as shown in Figure 3. The approach involves six basic steps:

5.1 DATA CODING

This step involves converting the problem from a n part, m machine, m processing step into a $n \times m$ part m machine problem, that is suitable for the proposed algorithm. Unique part ids and process ids are assigned to each of the $n \times m$ entities. This forms the primary key that uniquely identifies each part and the processing step that it needs to undergo. Alphanumeric coding scheme suggested by (Pongcharoen et al, 2003) has been used in this work as it can be extended easily and interpretation is straightforward. It may be noted that each of the initial n parts can have unique process plans with any number of processing steps. The assumption of m processing steps per part has been taken for simplicity and clarity in expression and demonstration.

5.2 DUE DATE ASSIGNMENT

This step involves assignment of due dates to each of the $n \times m$ entities. Due dates of the m^{th} processing steps for all the n parts are fixed (or predetermined). Due dates for the remaining parts can be assigned using different heuristics depending upon the goals and constraints of the shop floor; the due dates can be uniformly spaced in the planning horizon, or assigned based on the allowed makespan for the part. It is essential for the due date assignment rule to not contradict the precedence constraints that exist in the process plans or because of other shop floor constraints such as setup. In this work, a common intermediate due date has been assigned to all the parts for illustration of the overall algorithm.

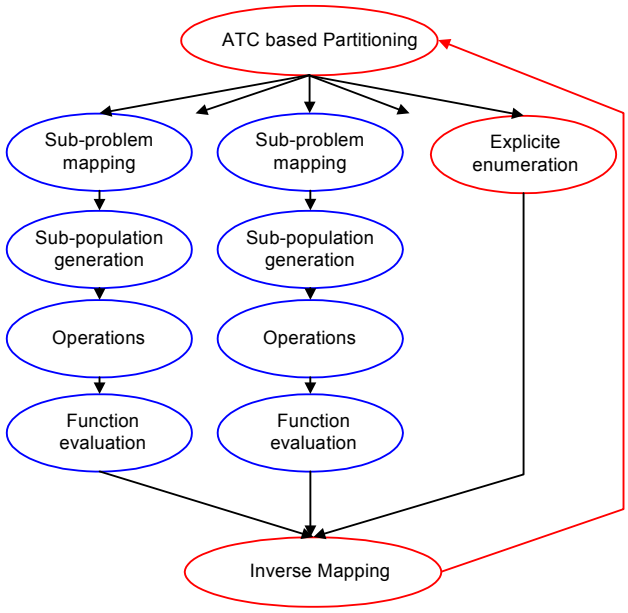

Figure 3. Hybrid Algorithm

5.3 ATC BASED PARTITIONING

Given the $n \times m$ entities, their processing times and due dates, and the shop floor conditions (constraints and machine states), the next step involves executing the ATC algorithm to partition the problem into active sub-problems. From Figure 2, it can be seen that the ATC converges in approximately 45 iterations. We fix $I = 50$ for this work. In general, we can take $I \approx 10T$, where T is the discrete time step used in Equation (7).

5.4 SUB-PROBLEM MAPPING

This step involves identification of the partitions and the sub-problems. Entities separated by an idle time of τ or greater duration in a machine schedule are assumed to be in different partitions. The value of τ has to be less than or equal to the smallest processing time of an entity in the active schedules separated by that τ. Once the partitions have been determined, mapping tables as shown in Tables 1 and 2 can be developed.

5.5 GA BASED SEQUENCING

This step involves creation of a sub-population of GA for the active sub-problems that have been created. To improve efficiency, sub-populations are not associated with all the active sub-problems. For a sub-problem α_i with $\ell_{\alpha i}$ entities, explicit enumeration involves $\ell_{\alpha i}!$ function evaluations. For the same size problem, GA based approach will require $N \times g$ function evaluations,

where N is the population size and g is the number of generations. Both N and g are dependent on $\ell_{\alpha i}$. Making conservative estimates and using heuristics mentioned in competent GA literature (Reed, 2000, 2002, Goldberg, 2002), the number of function evaluations can be approximated to $2\ell_{\alpha i} \times 2.8\ell_{\alpha i}$. Sub-populations are created only when the number of function evaluations using explicit enumeration far exceeds the number of function evaluations required by GA, i.e., $2\ell_{\alpha i} \times 2.8\ell_{\alpha i} \times 10 \leq \ell_{\alpha i}!$. This happens when $\ell_{\alpha i} \geq 7$. It the number of entities in an active sub-problem is greater than 7, sub-populations are created. The details of this step have been described in detail in Section 4.

Table 1 Mapping Table for an active schedule 1

Serial	Part id	t_{ij}	d_{ij}
1	14.2	9	320
2	6.2	8	320
3	1.2	7	320
4	4.2	6	320
5	3.2	5	320
6	2.2	6	320
7	11.2	7	320
8	7.2	8	320
9	31.2	9	320

Table 2 Mapping Table for an active schedule 2

Serial	Part id	t_{ij}	d_{ij}
1	2.1	5	150
2	3.1	4	150
3	31.1	3	150
4	4.1	2	150
5	7.1	1	150
6	11.1	2	150
7	14.1	3	150
8	6.1	4	150
9	1.1	5	150

5.6 INVERSE MAPPING

After a fixed number of iterations/generations/time, the best solutions in each of the sub-population is taken and reinserted into the ATC model. Infeasibilities, if any, are eliminated.

6. RESULTS

6.1 PROBLEM DEFINITION

The problem taken up to illustrate the application of the hybrid approach comprises of 9 parts with 2 processing steps each, and to be processed on 2 machines. The common due date for all the parts is 320 time units. The processing times and the Part ids (primary key) are presented in Table 3. The desired schedule should minimize the deviations from the due dates. The common due date problem has been selected because the solution lies in the discontinuous region and ATC is inefficient in converging to a solution.

Table 3. Problem Definition

Serial	1	2	3	4	5	6	7	8	9
Id	1	2	3	4	6	7	11	14	31
t_{ij}	5	5	4	2	4	1	2	3	3
t_{ij}	7	6	5	6	8	8	7	9	9
d_{ij}	320	320	320	320	320	320	320	320	320

6.2 ATC BASED PARTITIONING

The inntial 9 part, 2 machine, 2 processing step problem (Gantt Chart shown in Figure 5) was converted to a 18 part 2 machine problem. The due dates and part Id. assigned to the intermediate steps have been presented in Tables 1 and 2. The Gantt chart of the modified problem is presented in Figure 6. ATC was then used to partition the solution space. The mean squared deviation (MSD) performance of the algorithm is presented in Figure 6. The solution space partitions are illustrated in Figure 7.

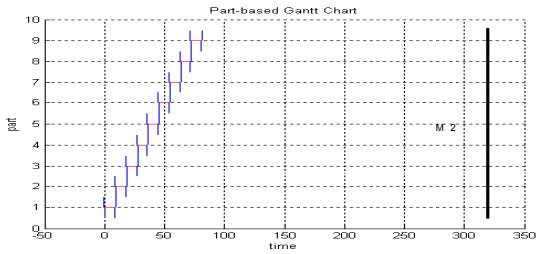

Figure 4. Initial Gantt Chart for Original Problem

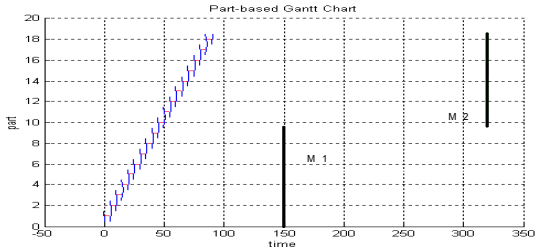

Figure 5. Initial Gantt Chart for Modified Problem

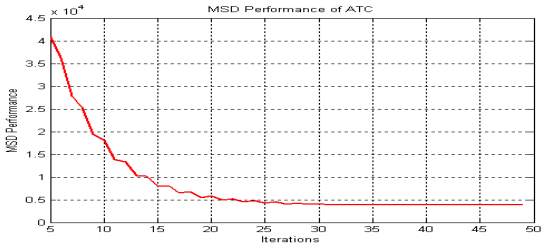

Figure 6. MSD performance of ATC

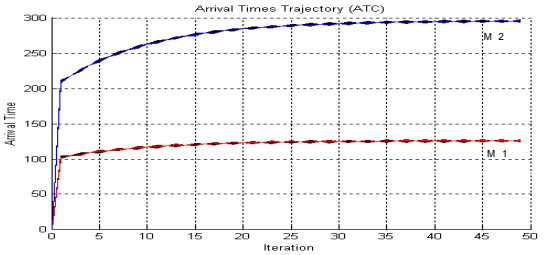

Figure 7. Solution Space Partitioning by ATC

The Gantt Charts created by the ATC algorithms are presented in Figure 8. The problem comprises of two local attractors and can be partitioned into two sub-problems. GA is then used in the sub-populations to

optimize the sequencing of the arrival times of the parts in each local sub-population.

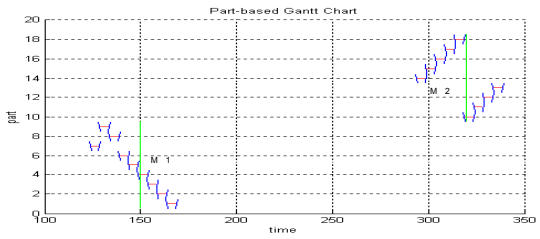

Figure 8. Final Gantt Chart for Modified Problem

6.3 GA BASED SEQUENCING

The mapping tables have been presented in Tables 1 and 2. These are used as inputs for the GA sub-populations. The representation and opertors presented in Section 5 were used for each sub-population thus created (in this case, 2). The parameters used in the sub-population are presented in Table 4. The MSD minimization performance in the sub-populations have been presented in Figure 9 and Figure 10.

Table 4. GA Paramteres

Id.	N	ℓ	$\eta_{init} \approx \ell$	$\eta_f \approx 2\ell$	λ	κ $(2\eta_f / \ell\lambda)$	Gen $\approx 2\eta_f$	p_m
1	9	9	10	20	2	8	40	0.1
2	9	9	10	20	2	8	40	0.1

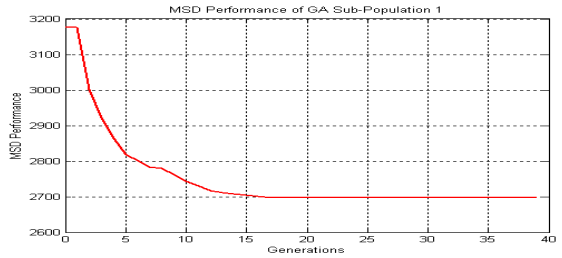

Figure 9. MSD performance of GA in Sub-Population 1

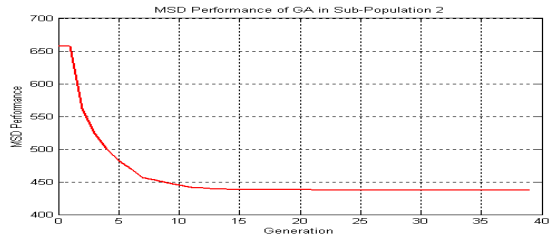

Figure 10. MSD performance of GA in Sub-Population 2

6.4 COMPARISON AND VERIFICATION

Initial results for the proposed hybrid approach have been presented in Table 5. There is an approximately 21% improvement from the solution that is given by using ATC alone, and 0.5% improvements from solution given by the heuristics for earliness-tardiness minimization heuristics (Ventura and Weng, 1995). Gantt chart displayed in Figure 11 shows the V-shaped alignment of the parts around the common due date as predicted by the heuristic developed by Ventura, 1995.

Table 5. MSD Performance of Algorithms

Algorithm	MSD
ATC (100 Iterations)	3840
V-Heuristic	3144
Hybrid	3005

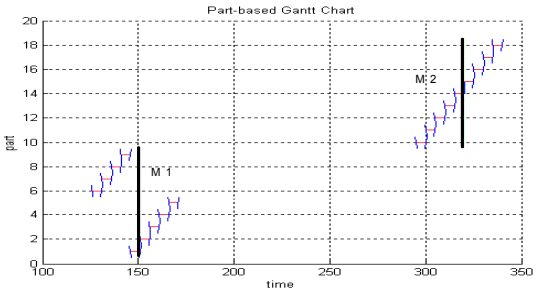

Figure 11. Hybrid Algorithm

7. CONCLUSIONS AND FUTURE WORK

ATC algorithm has an exponential convergence rate towards local basins of attraction in due date based scheduling problems. However, ATC is oblivious to optimal. The proposed hybrid algorithm introduces a global system view in the ATC algorithm by using GAs. ATC is used to partition the problem and generate initial populations near the local attractors. GAs are then applied within these partitions on the populations of solutions to seek superior solutions adaptively.

GA is efficient in the discontinuous region and blends well with the distributed architecture that ATC maintains. Local populations of chromosomes are generated for each individual zone of discontinuity, thereby limiting the length of the chromosome. For a n part m machine, m processing step job shop scheduling problem, a conservative approximation for the required number of function evaluations for the proposed approach is order $O(K(n\times m\times I)+(K\times m)\times(n\times m)^2) << O((n\times m^2)^2)$ and $O((n\times m^2)^2) << O((n!)^m)$. Offline analysis is conducted and the best solution is archived for generating the final schedule.

Other advantasge of the proposed hybrid approach are:

- Scalability: GA is efficient in the discontinuous region and blends well with the distributed architecture that ATC maintains. The computations for ATC can be executed on a separate processor in parallel, making them inherently suitable for massively parallel/distributed computing.
- Robustness and Fault tolerance: This feature is inherited from GA where the operations are on

populations of points rather than at single points in the search space.

Some aspects that are currently being explored for enhancing the hybrid approach are:

- A due date assignment strategy for the intermediate stages of a part. This will probably depend on the secondary objectives and constraints related to makespan. The assigned due dates for the intermediate stages dictates the distance between the partitions and the number of sub-problems.
- Time continuum, choice between the Baldwinian and the Lamarckian evolution, etc need to be considered in details.
- Improved synchronization strategies for ATC and GA communication. This may be related to the selected evolution strategy.

REFERENCE

Baker, K., "Introduction to Sequencing and Scheduling", New York, Wiley, 1974.

Cheng, T.C.E., and Gupta, M.C., "Survey of scheduling research involving due date determination decisions", European Journal of Operations Research, v38, p156-166, 1989.

Cho, S. and Prabhu, V.V. "A Vector Space Model for Variance Reduction in Single Machine Scheduling," *IIE Transactions*, Vol. 34, No. 11, pp 933–952, November 2002.

Brown, D.E., and Scherer, W.T, "Intelligent Scheduling Systems", Operations Research /Computer Science Interfaces, 1994.

Cochran, J.K., Horng, S.M., and Fowler, J.W., "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machined", Computers and Operations Research, In press article, 2002.

Lee, C.Y., and Choi, J.Y., "A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights", Computers and Operations Research, v22, n8, p857-869, 1995.

Lee, C.Y., and Kim, S.J., "Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights", Computers in Industrial Engineering, v28, n2, p231-243, 1995.

Li., Y., Ip, W.H., and Wang, D.W., "Genetic algorithm approach to earliness and tardiness production scheduling and planning problem", International Journal of Production Economics, v54, p65-76, 1998.

Lozano, J.A., Larranaga, P., Grana, M., and Albizuri, F.X., "Genetic algorithms: bridging the convergence gap", Theoretical Computer Science, v229, p11-22, 1999.

Gershwin, S.B., "Hierarchical Flow Control: A Framework for Scheduling and Planning Discrete Events in Manufacturing Systems", Proceedings of the IEEE 77, 1989, pp195-206.

Goldberg, D.E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Reading, MA: Addison Wesley, 1989.

Goldberg, D.E., "The Design of Evolution", Kluwer Academic Publishers, 2002.

Holland, J., "Adaption is Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press, 1975.

Kogan, K., and Khmelnitsky, E., "Scheduling: Control-Based Theory and Polynomial-Time Algorithms", Kluwer Academic Publishers, 2001.

Montana, D., Brinn, M., Moore, S., and Bidwell, G. "Genetic Algorithms for Complex, Real-Time Scheduling", IEEE Conference on Systems, Man, and Cybernetics, 1998.

Morton, T.E., and Pentico, D.W., "Heuristic scheduling Systems with applications to production systems and project management", Wiley Series in Engineering and Technology Management, 1993.

Pinedo, M., "Scheduling Theory, Algorithms, and Systems", Prentice-Hall, Inc, 1995.

Pongcharoen, P., Hicks, C., and Braiden, P.M., "The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure", European Journal of Operations Research, (In Press), 2003.

Prabhu, V.V., "Distributed Control Algorithms for Scalable Decision Making from Sensors to Suppliers," in *Scalable Enterprise Systems: An Introduction to Recent Advances* (Eds V. Prabhu, S.Kumara, M. Kamath), Kluwer Academic Press, to be published in 2003.

Reed, P., B. S. Minsker, and D. E. Goldberg, Designing a competent simple genetic algorithm for search and optimization, Water Resources Research, 36(12), 3757-3761, 2000.

Reed, P., and Minsker, B.S., Making Genetic Algorithms Work in the Real World: Guidelines from Competent GA Theory, in the Genetic and Evolutionary Computation Conference (GECCO) Tutorial Program, Erick Cantu-Paz (ed.), New York, NY, 2002

Rodammer, F.A., and White, K.P., "A recent survey of production scheduling" IEEE transactions on Systems, man, and cybernetics, v 18, n 6, pp 841-851, 1988.

Ovacik, I.M., and Uzsoy, R., "Decomposition methods for complex factory scheduling problems", Kluwer Academic Publishers, 1997.

Ventura, J.A., and Weng, M.X., "Minimizing the Single-Machine Completion Time Variance", Management Science, 1995.