

White Paper



Insights into Implementing

Genetic Algorithm based Production Schedulers

Swanand Sahasrabuddhe

Abstract

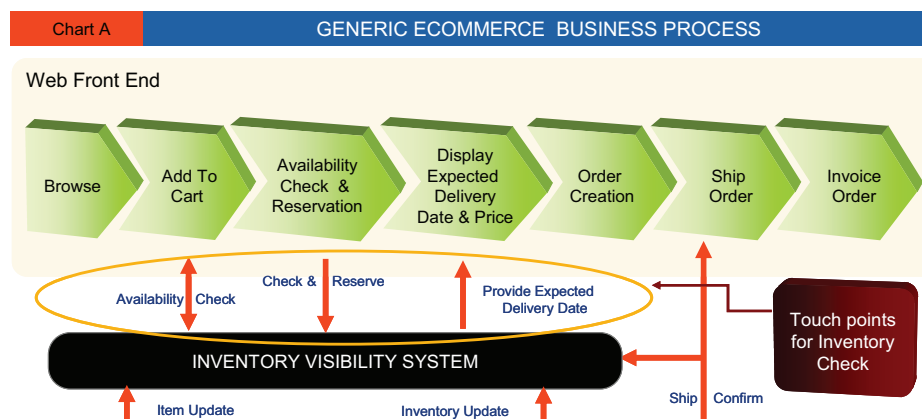
Production scheduling is part of the planning process, which occurs at the lowest level of time, product and location dimensions. Accuracy, optimal solution and response time for any changes to the input variables (planning lead time) can drive huge benefits at the shop floor level, resulting in reduced inventories, better fill rates and lower cost of production. There are many software products for scheduling shop floor production, based on various types of algorithms. One of such algorithms, Genetic Algorithm (GA) is increasingly becoming popular due to the benefits it brings. But at the same time, there are certain challenges that consultants face while implementing such GA based solutions. This paper provides insights into challenges in implementation of GA based solution and an approach to address them.

Introduction

It is often observed that the algorithms on which a particular optimizer solution is based, has certain inherent assumption. Therefore it functions well within certain boundaries or only under certain conditions. This document takes a look at how GA based systems are used to solve scheduling problems and how to use them appropriately.

Background – Production Scheduling in SCM

The place of production scheduling in supply chain planning is at the finest granularity in time, material and location. Often Production schedulers work on minute-to-minute basis in time, resource-to-resource and item-to-item basis in location/ product dimension of supply chains. Supply chain management starts at the highest level of formulating supply chain strategy. Based on the nature of the business and the product mix, an organization decides whether a responsive or a functional supply chain is appropriate, or a combination of these. The decisions regarding product design, sourcing and procurement, the product lifecycle management, replenishment planning, design of supply chain networks, distribution networks, etc. are taken accordingly. Demand planning generally gives the first trigger in form of demand forecast to start the supply chain planning. Supply chain planning then broadly decides when and where to procure from, produce, stock and distribute based on supply chain constraints, forecasted demand, margins across products, geographies, etc. Based on these recommendations, the planning of production and material procurement at factory level takes place. Although factory planning is at item level, it still works at a slightly aggregate level in terms of time dimension. In many of the manufacturing scenarios, it is required to plan a detailed schedule of production to optimize the use of limited resource capacities. The optimal planning at this level is often critical to satisfy supply chain plans at higher levels. Proper scheduling of production makes factories more agile and lean. The production scheduling plays a critical role in making supply chains more efficient and responsive, in situations where manufacturing environment is characterized by:



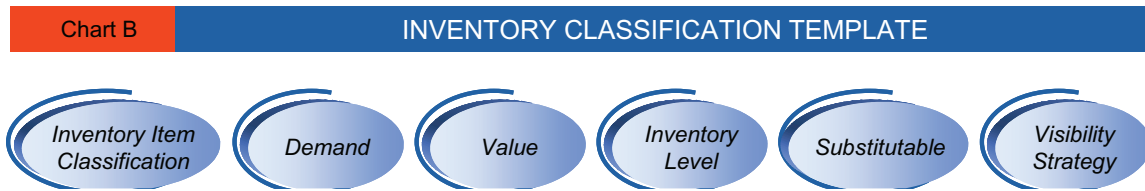
Some of the examples of such manufacturing environments are Paint Manufacturing, Automobiles, Cement Products, Steel Mills, Plastic Products, etc.

Production scheduling is often quite a complex task as it involves many SKUs to be scheduled across many resources, so as to minimize changeovers and down time and maximize the throughput. Hence Production Scheduling IT Systems are often used to get optimal schedules.

There are various approaches used to arrive at the optimal solution viz. Linear Programming, Integer Programming, etc. Algorithm, which is known as Genetic Algorithm, is also used by some of such Scheduling Optimizers. The inherent assumption in Genetic Algorithm approach is that *solution to scheduling problems is non-deterministic and hence the best way to solve such problems is to get as near as possible to the 'optimal solution'.*

What is Genetic Algorithm?

Genetic Algorithm (GA) is a heuristic based on the same strategy which according to the Darwinian Theory of evolution; nature has used to manifest itself in present form – which is ‘survival of the fittest’. Accordingly, this algorithm uses the following principles:



These concepts, when applied to a mathematical optimization problem, result in an optimal or the ‘fittest’ solution.

A chromosome in genetics is represented by one possible solution of the given mathematical problem. A set of chromosomes gives us a population, which is represented by a ‘set’ of possible solutions. The solutions placed ‘best’ in the population are selected for crossover process, which results in a new population. If this set has a better chance of survival, i.e., if the penalty value associated with this set is less than the previous set, then it is retained and used for next crossover. The assumption behind this procedure is that, there are sequences of parent chromosomes, which have better survival value, and a combination of such sequences would result into a child solution – hopefully, with a higher survival value. Many of the offspring solutions will not be ‘fitter’ than parents, but a few solutions will be. This procedure is repeated again and again till we get minimal improvements from successive runs. This approach has a tendency to fall into local optimal solutions, as the child solutions are more or less similar to ‘good parts’ of parent chromosomes. This problem is tackled again by using nature’s strategy of ‘mutation’. The mutation ensures the necessary randomization effects. A is more fit than B. A is fitter than B.

Thus, the basic steps that are followed in a generic GA process are ^[1]

1. Generate random population of feasible solutions for the problem
2. Evaluate the fitness (least penalty value/ highest reward value) of each solution
3. Create a new population by repeating the following steps until the new population is complete:
 - a. Select two parent solutions from a population based on their fitness (better the fitness, better are the chance to be selected)
 - b. With a crossover probability cross over the parents to form new offspring solutions.
 - c. With a mutation probability mutate new offspring solution at each position in solution.
 - d. Place new offspring in the new population
4. Use new generated population for a further run of the algorithm
5. If the end condition is satisfied, stop and return the best solution in current population
6. Go to step 2

The few important parameters, which can be manipulated to get a desired result out of GA, are:

1. Crossover probability: This indicates how often the crossover will be performed. If the crossover probability is more, then more offspring are made by the crossover and vice versa. If it is zero then the new generation is made from exact copies of chromosomes from old population.
2. Mutation probability: This indicates how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. Higher the mutation probability, higher is the number of times chromosomes are mutated and vice versa. Mutation avoids local optima.
3. Population size: This indicates how many chromosomes are in population. If there are too few chromosomes, only a small solution space is explored for optimum solution. Also, if there are too many chromosomes, the algorithm may slow down.

Why Genetic Algorithm ²

For problems with high degree of complexity, in terms of solution possibilities and constraints, GA works quite well as it does not try to find the exact solution, but rather tries to find the approximate of optimal solution.

GA has a number of advantages. It can quickly evaluate a large solution area. Bad proposals are discarded from the consideration set and hence do not affect the end solution negatively. GA doesn't have to know any rules of the problem - it works on its own internal rules. This is very useful for complex or loosely defined problems. Due to these advantages, many production scheduling optimizers use genetic algorithms.

Limitations of Genetic Algorithm ²

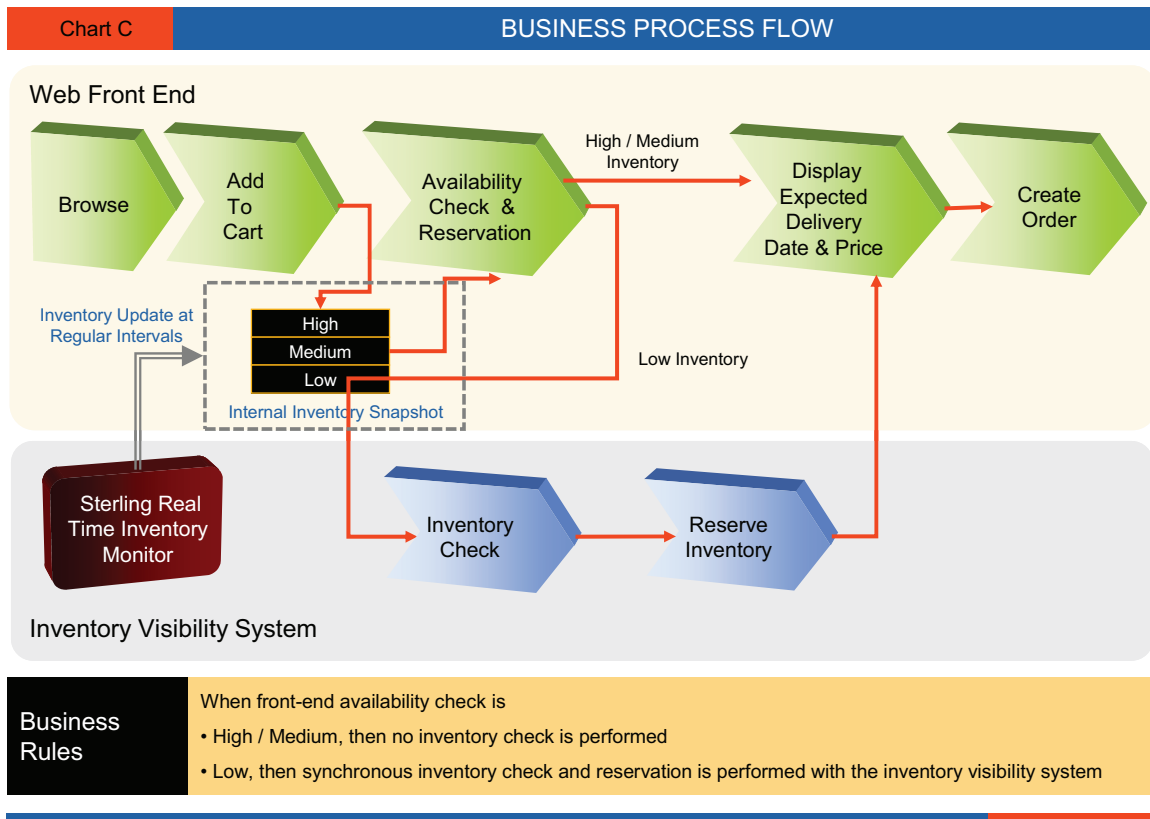
1. The fact that GA finds a solution through evolution becomes its biggest disadvantage. It does not evolve toward a good solution, but evolves away from a bad solution. This can cause a solution to evolve into a dead end. GAs risk finding a suboptimal solution or get caught in local optima trap, if not modeled correctly.
2. Since GA does not evolve toward a good solution, but evolve away from bad solutions, the number of iterations it takes to arrive at near optimal solution is very high, if the problem is not modeled correctly.
3. It is very difficult to get an optimal solution in very complex scenarios. Any algorithm, for that matter, is inadequate to give a 'best' result with the present level of computing resources available with organizations.

Some of these limitations can be overcome by appropriate use of a GA-based solution. The following considerations can help consultants to modify their approach in configuring GA-based solutions to get better results from scheduling systems, resulting in a successful implementation.

1. Understand the business requirements clearly. Try to prune all the unnecessary requirement judiciously.
2. It is advisable to use 'seed' solution to start off the optimization process. The seed solution should be based on the experience of the plant schedulers and should reflect business priorities.
3. Consider breaking down the optimization process into steps, whenever required. Trying to schedule and optimize all the manufacturing resources at the same time involves evaluation of very large number of possible solutions, out of which a large fraction is way below optimum, results in wastage of computational resources and high run times.
4. Be very clear about which constraints should be modeled as hard ones and soft ones. While assigning penalties to soft constraints, always try to relate those with a common evaluation factor such as monetary value associated with violation of those penalties. It is difficult to decide penalty values for entirely different types of constraints. Try to link different sets of constraints through proper penalty values. One way to do this is by assigning penalty values, based on economics associated with violations of these penalties. Although it is very difficult to arrive at a common denominator, it is quite a useful exercise to evaluate the importance of the different types of penalties with respect to each other in terms of monetary value.
5. Setting user expectations about the output of Scheduling solution is often critical. It is often found that there could be some local sub-optimal outputs in the final solution, which make the solution look 'bad'. But, most of the time, it is a result of the wrong settings of penalties and can be fine-tuned in the course of usage of the system.
6. In the case of use of seed solution and step-by-step optimization, it is advisable to use a small value for Mutation Probability. This is because if the seed solution itself is quite near to the perceived optimal solution, there is no point in mutating it and to go outside that near optimal area.
7. If required, the optimizer should be customized to solve a specific business problem.

Using GA-based Scheduling Solution – A Case

In this section, a brief account is given of the Infosys experience in implementing GA-based production scheduling optimizer for a Cement Products manufacturing company. This section discusses how above-mentioned considerations are used to get the best possible scheduling scenario.



Challenges in Scheduler Solution Implementation and Approach

The following section talks about the challenges faced in GA-based production scheduling systems implementation and the approaches taken to address those.

1. Problem Modeling

- a. **Resource Modeling:** It is often observed that just because a system can handle complex scenarios and large number of resources, items, etc., there is a tendency to model everything in the system. Here consultants and planners should judiciously decide modeling elements that are absolutely critical and the ones which can be taken out from the model.

In the above case, the rod mill was not at all a constrained resource and also its scheduling was not dependent on the type of end product being processed. So there is no point in modeling such resource just to get the start time of mill, from mill processing time and start time of the subsequent Mixing operation. It unnecessarily taxes the optimizer without adding any value.

- b. **Constraint Modeling:** All changeovers are not the same. A changeover from white to yellow color might consume fewer resources than from black to white. In most cases, these things are considered, but one should try to attach a monetary value to the penalties assigned to such changeovers. This makes all the penalty values have a common basis, and thus are more authentic reflection of the business reality.

One can use monetary value for similar constraints for assigning penalty values, as it is easy to have a common basis. But the most difficult part is to compare penalty values of different types of constraints. How do you weigh a changeover constraint against a resource choice constraint?

For example, resource choice constraints are given penalty values 1, 2, 3, etc. - based on the order of preference and on the other hand, the machine changeover constraint can have values - say 100, 200, etc. Independently, such values should not cause any problem. But when there is a choice to be made between which of these two constraints to violate, the optimizer will always give preference to the changeover constraint, even if it means scheduling the tasks on bad resource choices. This might result in lower throughput or bad quality of product.

One way to avoid such problem is to have a common denominator to assign penalty values, which could be as stated previously, an economic consideration. In this example, if the 'bad' resource choice lowers the throughput by 10% (say due to lower production rate of the resource), it will result in reduction in contribution to the bottom line. Normally, planners are aware of the margins of each product. Similarly, one can know the resource requirement for a changeover, downtime for that changeover, and thus subsequent production loss, and therefore can attach a monetary value to this changeover. Now, one can compare these two on the basis of a common parameter, and thus assign appropriate penalty values. This approach might be quite complicated sometimes, as it is difficult to perform such an analysis for all the products and for all the resources. But, this can be done for major products and resources and then similar values can be assigned to remaining products.

2. Run Time and Sub-optimal solutions

GA by its nature requires large number of iterations to arrive at a good solution. And more the complexity of the scheduling scenario or the elements modeled, more is the time required to get a feasible solution and the time required for evaluation of each solution. It is suggested to give a good starting point to the optimizer. This starting point, called a 'seed' solution should be based on the planners' experience and knowledge about what a good solution should be.

In above example, the seed solution approach can be taken. All the tasks to be scheduled can be given certain 'names', which are basically concatenation of various attributes of the item to be scheduled. The sequence in which the concatenation is done is determined by the contribution margins and other parameters. For example, if all similar length products are to be scheduled together and the length is always considered over breadth, thickness or color, then the concatenated 'name' should have length appearing before breadth. All the attributes are properly arranged and then the tasks are sorted based on the 'names', and thus this seed query generates a sequence of task in some order that forms the starting point of the optimizer search.

3. Use of step-by-step optimization

In few cases, some tasks are scheduled at some predetermined time. Sometimes it is good to break up a large and complex routing into smaller routings and schedule these in the order of their place in parent routing.

In the above case, the complete optimizer run can be broken down in 5 steps. The first step schedules Mixing operations for some type of products, which are to be scheduled on some predetermined resource, regardless of the remaining schedule. Thus, the first step would run for only a single iteration, which reduces the run time significantly. This also reduces the number of mixing tasks to be scheduled in second step. Second step schedules the remaining mixing tasks. The third step schedules all the molding tasks, which anyway are to be scheduled only after mixing tasks. Thus, having a separate step for these tasks, which are dependent on mixing tasks, and running it only after mixing steps makes the problem definition and population definition for the optimizer less complex. The fourth step is for all other remaining tasks like curing, cutting, inspection etc., which come after molding, but before coating operation. The final step schedules all the coating tasks. Thus, optimization is broken down into steps with a focus on optimizing critical operations and resources first, and then the subsequent operations. One of the drawbacks of using this approach is that this could lead to local-suboptimal solution. But, one has to evaluate other parameters - like run time, quality of solution without this approach - and then decide how many steps to use and how to break down the optimization process.

4. Understanding the Output and Fine Tuning

Even after proper assignment of penalty values, the output schedule sometimes might not look 'good'. But it must be understood and made clear right at the stage of setting user expectations that this could be result of complex interaction of hundreds of constraints, which often are not assigned proper penalty values. Thus this can be avoided though continuous fine-tuning of the model.

It is also critical to explain to users why something is modeled in some particular way, to make them understand and appreciate the complexity involved in use of GA in scheduling. For example, the users can misunderstand the use of steps in scheduling and only single resultant iteration, in some steps, as sub-optimal use of the optimizer capabilities. Use of predetermined sequence in seed solution can also be misunderstood to be hard coding of the schedule. But, a detailed explanation on GA fundamentals and how the use of seed solution can greatly reduce run times and also give near optimal solution can help them understand the GA working and appreciate it.

5. Customization

Every factory is unique in terms of scheduling rules, requirement, etc. and it's almost impossible to make a generic solution that fits all. Therefore, Production Scheduler Systems implementation often involves a lot of customizations. So, it is very important to follow a proper methodology for any such development, right from the beginning.

Conclusion

Using Genetic Algorithms has its advantages and challenges. A proper analysis of the requirement priorities, constraints, models and objective function helps in overcoming the challenges for a successful implementation. This paper discusses the suggestions that consultants should consider while modeling GA-based scheduling solutions.

References and Further Readings on GA

1. <http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php>
2. <http://subsimple.com/genealgo.asp>
3. http://en.wikipedia.org/wiki/Genetic_algorithm
4. <http://lancet.mit.edu/~mbwall/presentations/IntroToGAs/P001.html>



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.