Optimizing Flow Shop Sequencing Through Simulation Optimization Using Evolutionary Methods

Sucharith Vanguri [1], Travis W. Hill [2], Allen G. Greenwood [1]

[1] Department of Industrial Engineering
260 McCain Engineering Building
Mississippi State University
Mississippi State, MS 39762

[2] Center for Advanced Vehicular Systems
153 Mississippi Parkway
Mississippi State University
Canton, MS 39046

Abstract

This paper describes an approach to use an adapted Evolution Strategies (ES) algorithm to generate improved sequences for producing unique parts in a flow shop. The algorithm uses principles from both genetic algorithms and Evolution Strategies. While several alternative algorithms were considered, the focus of this paper is on the one that performed the best for this problem domain. The best algorithm is an ES that implements a new mapping technique (Genotype-Phenotype) to convert its real-valued gene representation into a valid job sequence. The approach also uses production heuristics to generate the initial set of sequences, thus providing a better starting point and accelerating the optimization process. The fitness of each sequence generated by the algorithm is evaluated by a discrete-event simulation model of the flow shop. The algorithm and simulation model are a part of a decision-support system that was developed to optimize ship panel construction at Northrop Grumman Ship Systems. A design-of-experiments approach is used to configure the efficiency of the algorithm for different problem sizes. This analysis helps select the optimal set of parameters. This paper includes details of routines and provides results from various optimization runs.

Keywords

Evolution Strategies, Flow shop Scheduling, Optimization

1. Introduction

The flow-shop scheduling problem (FSP) is a well-known problem found in many manufacturing applications. The FSP is known to be NP-Complete for relatively large, *n* job, *m* machine problems. Due to its complexity it is necessary to use heuristic approaches to solve the problem. Algorithms and heuristics developed to solve flow shop problems include Johnson's rule (for a 2 machine problem), tabu search, simulated annealing and genetic algorithms. Evolutionary computing (EC) is widely used in similar combinatorial problems; ample contributions are present in literature for solving the traveling salesman problem (TSP), a similar combinatorial problem. However, only few EC solution approaches for solving FSP have been developed. This paper presents an adapted Evolutionary Strategies (ES) solution approach to the FSP. This optimization algorithm is part of an elaborate decision support system (DSS) developed for Northrop Grumman Ship Systems. A panel shop responsible for manufacturing components for ships is considered for experimentation and the actual workflow is evaluated using a discrete-event simulation model of the shop. The following sections provide details on the ES approach, implementation, and results.

2. Evolutionary Computation Methods

Evolutionary computation methods are heuristics that seek optimum solutions by mimicking biological evolution processes. These methods use the *survival of the fittest* principle to promote evolution over generations. Each generation consists of individuals, collectively known as a population. Each individual is a representation of the parameters related to the fitness function. The representation of an individual is called a chromosome and is analogous to chromosomes present in living cells that control characteristics of an organism. The best individuals in a population are selected as parents to breed the next generation. The new population members compete within themselves and the best individuals are selected again. This process of selection combined with sporadic mutation contributes to the progression of the population towards the global optimum. Evolutionary computation is a broad

term that envelops all methods mimicking evolution; different methods have been proposed and developed over the years. Genetic Algorithm (GA) and Evolution Strategies (ES) are just two of the more popular methods.

The GA is a method that uses binary values in its chromosome to represent the variable values; it was proposed by Holland [1] and further developed by Goldberg [2]. The GA uses crossover and mutation operators to create a new set of individuals. GAs though effective, lack speed and efficiency when compared to more recent methods. A GA requires very large population sizes and the binary chromosome representation increases the length, thereby reducing efficiency. Crossover is considered to be the major evolution factor in the GA with mutation preventing the algorithm from being trapped in a local optimum. Various adaptations have been developed to solve specific problems, such as the TSP, and some adaptations have integer representations with special crossover and mutation operators.

The ES proposed by Schewfel [3, 4] is based on a real-value representation of variables and is generally used to optimize mathematical functions. An ES operates using real-value objective function variables by applying recombination (crossover) and mutation [6]. A solution is represented in a gene format, consisting of function variables and strategy parameters. An ES defines and uses a strategy parameter for every variable present in the objective function. The gene resembles an array of size $2n$, where n is the number of variables in the objective function that need to be optimally determined. The first set of n values corresponds to the variables and the next n values form the strategy parameters. Function variables represent the solution, whereas the strategy parameters help the algorithm traverse the solution space more effectively by mutating function variables. The mutation is a simple neighborhood search process where the given variable value is modified by adding or subtracting a normal variate based on the variable value and the strategy parameter. Mutation plays the main role in the ES optimization process to improve the population and avoid being trapped in local optima. In general, ES are considered to be more efficient and faster than GA implementations and work with small population sizes.

In an EC algorithm each gene is associated with a fitness value; the value is evaluated by using the variable values from the gene and substituting it in the objective function. The EC operates with just the evaluated value and has no knowledge of the objective function. Thus any problem with configurable parameters can be optimized using an EC approach. A fitness-based selection process is used, where all the population members are sorted based on their fitness values and the best members are selected to be the parents for the next generation. The best is dependent on the minimization or maximization goal. Other more complex tournament-type selections can be used where every individual in the population competes with randomly-chosen individuals and population members are selected based on the number of wins. The selection process also involves a critical decision, i.e., elitism, which influences the behavior of the algorithm. Elitism, in general, means giving preference to higher individuals; in evolutionary terminology it means giving preference to the best individuals regardless of its age. This enables the parents to compete with the offspring and have a chance for survival. If elitism is allowed in an EC, a good solution will survive through generations and will be selected to produce more offspring. However elitism can also lead to an early convergence to local optima. In ES terminology, elitism is denoted by the notation $(\mu+\lambda)$ where μ denotes the number of parent solutions and λ denotes the number of offspring. The '+' represents that the selection process will consider both parents and offspring. The non-elitist strategy is represented by (μ,λ).

Both the GA and ES have been used in different problem domains and have established their significance in the field of optimization. Several modifications of the basic approaches have been proposed and implemented to address specific problems. Large combinatorial problems, such as the FSP and TSP, are routinely solved using modified GA methods. On the other hand, ES methods have limited application in solving combinatorial problems. In one case, Rudolph [5] uses a mapping technique for applying ES to solve a TSP problem, but no research could be found where an ES was used to solve a FSP problem. Even though TSP and FSP are different problem formulations with different constraints, they appear identical from the EC perspective. Both problems, when formulated for any EC method, essentially represent a non-repetitive integer sequence, which forms either the travel route or the job sequence. This similarity lead to the adaptation of an ES to solve the FSP problem, details of this implementation are provided in the next section.

3. Optimization Approach

The objective of a flow shop problem is to find the best sequence in which to process jobs through the flow shop. "Best" is evaluated in terms of the throughput of the shop; i.e., produce a given set of jobs through the flow shop in

the shortest time possible, given the operational, physical, and programmatic restrictions. The jobs need to be processed on a single series of machines and are processed in the same order as they enter the system. Personnel and machine resources and space limitations control the pace of production.

An ES algorithm is employed to "optimize" the flow shop-sequencing problem. The core algorithm is described in the following pseudo code:

1. START
2. Generate initial population
   (a) Randomly generate values for variables
   (b) Initialize strategy parameters for each variable
   (c) Evaluate initial population
3. Generate next generation
   (a) Select two parent members at random
   (b) Create an offspring solution by selecting each variable value from either of the parent's corresponding variable (discrete recombination)
   (c) Define strategy parameters for the offspring as an average of the corresponding two parent values
   (d) Mutate strategy parameters
   (e) Mutate each variable of the offspring based on the new strategy parameter
4. Evaluate the objective function
5. Sort the solutions
6. Repeat from Steps 3 through 5 until the desired tolerance is achieved
7. END

The basic ES algorithm needs to be modified in order to implement it on an FSP. Modifications include a mapping scheme, fitness function, seeding the initial population, and parameter adjustments. The behavior of the algorithm can be controlled by adjusting a set of parameters including: initial strategy parameter values, parent to offspring ratio, elitism, maximum number of generations and tolerance. The stopping criterion is either the maximum number of generations or a tolerance that is defined as the difference between the best solution fitness and the average fitness of parent solutions. All parameters are accessed through a configuration file, thereby enabling the end user to calibrate the algorithm as required for different problems. This section provides details on these modifications.

Since the FSP involves sequences of non-repetitive integers that are used to represent a job sequence, a new form of representation or translation is required. The GA chromosome can be easily modified to be represented as a set of integers; however, a similar approach to the ES is not possible. Rudolph [6] solved a TSP by obtaining a valid sequence from a set of real numbers using the idea of Genotype and Phenotype mapping to develop an encoding and decoding process. Phenotype can be defined as the observable traits of an organism which are caused by a specific gene combination or structure called the genotype. In simpler terms, the phenotype is all that can be seen and the genotype is the more complex structure well hidden in the organism that controls the characteristics. Since the fitness function is related to behavior, a proper genotype to phenotype mapping will ensure that an accurate fitness evaluation is made. The adaptation of the ES lies in the encoding scheme which allows the ES to operate on its real values while being translated to relevant information for the actual problem. The solution or chromosome values are not altered in this process, a new array of numbers corresponding to the sequence is generated which is passed out of the algorithm for fitness evaluation. Once the fitness values for all solutions are found they are applied back to the population and then sorted for the selection process.

Encoding is achieved by assigning randomly-generated numbers [0,1) to the given sequence after sorting. The assignment ensures that the lowest number is assigned to the lowest job identifier, and proceeds until all integers are mapped. As shown in the example in Table 1, a set of random numbers (0.482, 0.002, 0.945, 0.205, 0.567) is sorted in ascending order (0.002, 0.205, 0.482, 0.567, 0.945) and assigned to the original set of integers according to rank. Since the algorithm is designed to pass the complete real-value solution set to the next generation, there is no need for any future encoding. Encoding is performed only for the initial parent population.

Table 1. Encoding example.

Input Sequence (Input)	5	1	2	4	3
Encoded Chromosome (Output)	0.945	0.002	0.205	0.567	0.482

Decoding of the chromosome is performed by assigning ranks to the set of real values present in the chromosome that represents the objective function variables. As shown in the small example in Table 2, the lowest number is assigned an index value 1. Index value 2 is assigned to the next lowest number and this process continues until all of the real values are numbered. If a tie in ranking exists, then a random assignment is made. This ranking process can be simplified by using the array index value after the real value set is sorted in ascending order. Ranking ensures that no duplicates are created. This encoding and decoding process does not disturb the original chromosome consisting of the variable and mutation parameters.

Table 2. Decoding example.

Chromosome Values (Input)	0.345	0.683	0.125	0.567	0.001
Decoded Sequence (Output)	3	5	2	4	1

An objective function value is associated with each gene, in the case of the FSP, this objective function value is the makespan (in days) to complete all jobs. Complex objective functions can also be defined by incorporating additional measures like tardiness. Due dates, if present, can be used to measure tardiness and added to the makespan for use by the optimizer as it attempts to minimize or maximize this objective function as required. The fitness values are used in the selection process, the basic fitness-based selection of and ES is retained. Another decision parameter, elitism, is to be defined for the selection process. Both $(\mu+\lambda)$ and (μ,λ) strategies are built into the algorithm and a configuration parameter that is defined by the user enables the selection.

Generally, initial population members are generated randomly in order to enable a wide range of solutions. However, a totally random start will delay the algorithm convergence. Therefore, in order to achieve faster improvement, the initial population is generated using established efficient heuristic scheduling rules. A similar process of seeding the initial population was also used by Reeves [6], where one of the random initial population members is replaced with a heuristic solution. Reeves employed the NEH algorithm from Nawaz, et al. [7] and showed that this seeding process significantly reduces the time to find an optimal solution. The seeding approach was adopted in the ES implementation where, common scheduling algorithms such as Shortest Processing Time (SPT), Earliest Due Date (EDD) and Critical Ratio (CR) are used for generating additional starting solutions. The SPT orders components according to their expected processing time. SPT is near optimal for finding the minimum total completion time and weighted completion time. The EDD provides a sequence based on component due dates and effectively reduces tardiness. The CR is the ratio of the time remaining to the work remaining; it works well on minimizing average lateness. The solutions provided by the heuristics may not be optimal, but provide a more efficient starting point for the search. More solutions than the required initial population can be provided as input to the optimizer algorithm, and then the algorithm will evaluate all solutions and select the required number of best solutions to form the initial population. This enables the user to employ similar heuristics to generate a broader set of solutions.

4. Implementation

The optimization algorithm is developed as part of a DSS for the panel shop that is considered to be the bottleneck for the shipyard [8]. The shop fabricates and assembles “panels” which are major components that are used in all ships. The panel shop primarily operates as an assembly line; each panel goes through the same sequence of processing steps, although the work content varies greatly at each step. The panels are composed of very large plates of steel which are unique and vary considerably in terms of their physical attributes and work content. The panel shop is essentially a flow shop; however the problem is highly complicated with the consideration of due dates, dynamic resources and high variability. A static model formulation will not truly represent the complexity of the system. However, a simulation model adequately captures the dynamics of the panel shop. The simulation model is an accurate representation of the original system consisting of all resources (people and machines) and operations logic and parameters that approximate the behavior of the shop. Each sequence generated by the ES optimization algorithm is processed by the discrete-event simulation model. The simulation model is used to evaluate the objective function. The number of days late and makespan are calculated from the simulation model output and are used by the optimizer as it attempts to optimize this objective function. It is observed that the optimizer primarily focuses on reducing tardiness. This behavior is attributed to the fact that the bottleneck of the panel shop is an automated process and there is little time variation possible in the makespan with changes in schedule. The DSS controls and coordinates the operation of the simulation model and the optimization algorithm.

It has also been observed that the algorithm has to be adjusted for achieving the best performance for different problem sizes. Since there are many configuration parameters that can be set at numerous levels, a perfect combination cannot be defined without further analysis. To study the effect of each of these parameters a design-of-experiments approach is suggested by Ruiz, Maroto and Alcaraz [9], where statistical analyses are performed to select the best combination of parameter values using results obtained from a quick limited run. The parameter values are then used for obtaining the global optimum, for the Taillard FSP problem set [10], by allowing the optimizer to run longer.

In order to provide a faster optimization algorithm a similar study is performed. A complete factorial design between the factors: number of parent solutions μ, parent-offspring ratio λ/μ, initial strategy parameter values, and elitist strategy, is evaluated. The simulation model is not used for this experiment, a simple code segment (math model) is employed to evaluate the makespan of a sequence for a given flow shop problem data. The math model functions as a simple flow shop and does not consider variability, due dates and other problem specific restrictions. Researchers have commonly used the Taillard FSP test set [10] for the evaluation of their algorithms. Selected problem instances from the same data set are used in this experiment. Three problem sets of 20, 100 and 200 jobs are used. A full factorial design is setup with 8x8x6x2 treatment combinations for the factors defined earlier respectively. The statistical analysis shows some critical interaction effects between the type of elitist strategy and the initial sigma values used. Only these two factors and their interaction significantly affect the optimization performance. In general the $(\mu+\lambda)$ strategy performed better than the (μ,λ) strategy. As the number of panels increases, a reduction in the initial strategy parameter values improved convergence. Table 3 shows suggested values of configuration parameters for different problem sizes. This experiment can be extended to derive a regression model where in the parameters can be defined from the problem data and then used in the optimizer.

Table 3. Configuration parameter suggestions based on experimental analysis

Number of Jobs	Initial strategy parameter	Elitist strategy	Number of parents	Parent – offspring ratio
20	0.05 to 0.1	(μ,λ)	>6	>6
100	0.001	(μ+λ)	>4	>7
200	0.001	(μ+λ)	>4	>5

5. Results

The algorithm is evaluated using six sets of panels of increasing quantity. The original schedule, Earliest Start Date (ESD), which the panel shop currently uses, is defined as the base schedule. This base schedule is used to generate three schedules using heuristics. Then each of these schedules is evaluated using the simulation model. The fitness value, a combination of makespan and the total tardiness, for each are collected and the optimizer is allowed to run for 100 generations. The optimizer disables variability and only evaluates one replication in the simulation model for each sequence during the optimization process. Once the best solution is determined, it is evaluated for multiple replications in the simulation model and the fitness value is collected. Table 4 presents the fitness values of the initial population and the final best solution, for 1 replication and 25 replications.

Table 4: Comparison of fitness between heuristics and ES algorithm.

	ESD (Base)		EDD		SPT		CR		OPT		Best : OPT %		Base : OPT %	
Reps/ Panels	**1**	**25**	**1**	**25**	**1**	**25**	**1**	**25**	**1**	**25**	**1**	**25**	**1**	**25**
20	1.61	1.25	1.43	1.15	1.61	1.25	1.68	1.33	1.00	1.00	29.99	13.30	37.91	19.89
50	1.79	1.19	1.47	1.19	1.79	1.19	2.31	1.79	1.00	1.00	31.90	15.66	44.02	16.19
100	1.80	1.59	1.16	1.01	1.79	1.59	2.65	2.18	1.00	1.00	13.64	1.04	44.47	37.05
150	1.73	1.64	1.11	1.02	1.75	1.62	2.89	2.84	1.00	1.00	9.50	1.66	42.07	39.12
200	2.03	1.80	1.23	1.14	1.92	1.77	2.95	2.57	1.00	1.00	18.98	11.99	50.73	44.50
250	1.96	1.70	1.14	1.06	1.84	1.65	2.73	2.32	1.00	1.00	12.65	5.71	49.09	41.06

The data has been normalized within each row. Data for both replications is presented in order to make a comparison between the final outcome of the ES and the initial solutions provided. The comparison has to be made between data of the same replication count. Variability plays a major role in increasing the make-span; this effect

reduces the quality of the fitness value when replicated. The Best : OPT and Base : OPT columns show the percentage improvement the optimizer achieved over the best of the initial solutions and the base solution respectively. It is obtained using the relationship (initial best – final best)/ final best.

As shown in Table 4, the ES optimizer outperformed all of the heuristics. A comparison between the heuristics shows that the EDD fares better than the base schedule and other heuristics. The trend in increasing improvement of the optimizer over the base case as the number of panels increase can be attributed to longer schedule time periods. The longer time periods may provide enough flexibility to reduce tardiness by changing the production sequence. EDD if employed will lead to better solutions as tardiness is part of the objective function. However to achieve higher improvements the optimization algorithm is recommended. The results shown are based on a single instance and hence do not represent a problem size. All schedules having 100 panels, for example, may not produce the same improvement results. This is primarily due to high variation in work content of the panels.

6. Conclusion

An evolution strategies algorithm is developed for solving a dynamic flow shop problem for a specific shipbuilding application. A real-valued ES algorithm is modified to use a mapping technique to produce valid job sequences and to use heuristic solutions as the initial population. This technique enables the optimization features of ES to be applied to a combinatorial problem. The optimization approach is developed, tested, and imbedded in a DSS for use in a real manufacturing shop problem. Statistical analyses are used to set different configuration parameters for the ES algorithm to achieve faster and better results. A full factorial design is used for this purpose. The algorithm through the DSS provided better schedules as an alternate to their existing ones. The DSS also provides a chance to evaluate any changes, using the model, before actual implementation. The optimizer is built in a generic, adaptable form, which enables it to be integrated and used with similar problems seamlessly. Further research includes experimental methods to simplify decision making in setting parameter values, provision of additional mutation and crossover options, and methods to improve convergence rates.

7. References

[1] Holland J. H., 1992, *Adaptation in Natural and Artificial Systems*: MIT Press.

[2] Goldberg D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning* Addison-Wesley Pub. Co.

[3] Beyer H. G. and Schwefel, H. P., 2002, "Evolution Strategies – a Comprehensive Introduction," *Natural Computing: an international journal* vol. 1, pp. 3-52.

[4] Schwefel H.-P., 1975, "Evolutionsstrategie Und Numerische Optimierung," TU Berlin, Germany.

[5] Rudolph G., 1991, "Global Optimization by Means of Distributed Evolution Strategies," presented at Proceedings of the 1st Workshop on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol. 496, pp. 209-213.

[6] Reeves C. R., 1995, "A Genetic Algorithm for Flowshop Sequencing," *Computers and Operations Research*, vol. 22, pp. 5-13.

[7] Nawaz M., Emscore Jr, E. E., and Ham, I., 1983, "A Heuristic Algorithm for the M-Machine, N-Job Flow Shop Sequencing Problem," *Omega*, vol. 11, pp. 91-95.

[8] Greenwood A. G., Hill, T. W., Miller, J. W., Vanguri, S., Eksioglu, B., Jain, P., and Walden, C. T., 2005, "Simulaton Optimization Decision Support System for Ship Panel Shop Operations," *Proceddings of the Winter Simulation Conference*.

[9] Ruiz R., Maroto, C., and Alcaraz, J., 2005, "Solving the Flowshop Scheduling Problem with Sequence Dependent Setup Times Using Advanced Metaheuristics," *European Journal of Operational Research*, vol. 165, pp. 34-54.

[10] Taillard E. D., 1993, "Benchmarks for Basic Scheduling Problems," *European Journal of Operational Research* vol. 64, pp. 278-285.

Acknowledgements

The authors acknowledge the significant contributions of Charles LaRue, Timothy Hardy, and Dr. Marino (Nick) Niccolai of Northrop Grumman Ship Systems. Their support and process knowledge were invaluable to the success of this project. This project was funded by Northrop Grumman Ship Systems, the Manufacturing Extension Partnership of Mississippi, and Mississippi State University.