Flow shop scheduling using enhanced differential evolution algorithm

Donald Davendra
Faculty of Applied Informatics
Tomas Bata University
Zlin, Czech Republic
davendra@fai.utb.cz

Godfrey Onwubolu
School of Engineering and Physics
University of the South Pacific
Suva, Fiji Islands
onwubolu_g@usp.ac.fj

KEYWORDS
Flow shop scheduling, differential evolution, heuristics.

ABSTRACT

This paper presents a new approach of differential evolution to scheduling optimization problem. The developed approach is viewed as an enhanced varient of differential evolution, incorporation new child correction schemas and coversion schemas from differential to discrete domain. The heuristic is extensively evaluated with the scheduling problem of flow shop and compared with published results.

INTRODUCTION

Metaheuristics are the common tool utilized to solve complex manufacturing problems. The advantage of this process is the production of viable results within the given constraints and resources. Flow shop scheduling (FSS) can be considered as one of the common manufacturing problems that is regurarly realized using optimization techniques. The evolution of optimization techniques has been mainly attributed to the increase in complexity of problems encountered. Two branches of heuristics exist: constructive and improvement (Onwubolu and Mutingi 1999). Constructive methods are usually problem dependent (Cambell *et al.* 1970, Nawaz *et al.* 1983). Improvement methods are those involving population-based heuristics which usually follow a naturally occurring paradigm. Some of these are genetic algorithms (GA), tabu search (TS), neural networks (NN), simulated annealing (SA) and particle swamp optimization (PSO) among others.

Differential evolution (DE) algorithm was introduced by Price and Storn (1999). Since then, due to its effectiveness, a lot of advanced work (see Onwubolu and Babu 2004; Lampinen and Storn 2004 and Lempinen and Zelinka 1999) have been conducted in order to realize the full potential of this viable approach.

In its canonical form, DE is designed to solve differential problems, which involve continuous values; that is, there is no discriminating feature in DE between values within a solution. This approach is effective; however a lot of problems involve solutions which are permutative, such as FSS. To achieve the desired heuristic, certain modifications have to be undertaken to change the operational domain of DE from continuous to discrete. Initial work has been done by Onwubolu and Davendra (2006), to transform the operational domain, however to improve the solutions further, enhancements were required. This varient was termed Discrete Differential Evolution (DDE).

This paper covers the work done to DDE to enhance it to enhanced differential evolution (EDE) algorithm, and its application to multiple FSS problems, in order to show its effectiveness over a wider range of FSS problems.

FLOW SHOP SCHEDULING

In many manufacturing and assembly facilities a number of operations have to be done on every job. Often, these operations have to be done on all jobs in the same order, which implies that the jobs have to follow the same route. The machines are assumed to be set up and the environment is referred to as flow shop (Pinedo 1995). The flow shop can be formatted generally by the sequencing on *n* jobs on *m* machines under the precedence condition. The general constraints that are assessed for a flow shop system is the time required to finish all jobs or makespan, minimizing of average flow time, and the maximizing the number of tardy jobs.

The minimization of completion time for a flow shop schedule is equivalent to minimizing the objective function

$$\Im = \sum_{j=1}^{n} C_{m,j} \tag{1}$$

where
$C_{m,j}$ is the completion time of job *j*. To calculate $C_{m,j}$ the recursive procedure is followed for any i^{th} machine j^{th} job as follows:

$$C_{m,j} = \max\left(C_{i-1,j}, C_{i,j-1}\right) + P_{i,j} \tag{2}$$

where $C_{i,j} = k$ (any given value) and $C_{i,j} = \sum_{k=1}^{j} C_{1,k}; C_{i,j} = \sum_{k=1}^{i} C_{k,1}$ which represents *i* as the machine number, *j* as the job in the sequence and $p_{i,j}$ as the processing time of job *j* on machine *i*.

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

ENHANCED DIFFERENTIAL EVOLUTION ALGORITHM

EDE is an extension of DE and DDE, and possesses the same novel approach that has made DE such a robust heuristic. EDE in addition to being an extension of DE, in regards to the inclusion of discrete optimization, is also an enhancement of DE. Enhancement is only archeived through the implementation of routines which improve the solution quality. The outline of EDE is given in Figure 1.

- **Initial Phase**
  1. *Population Generation:* An initial number of discrete trial solutions are generated for the initial population.
- **Conversion**
  2. *Discrete to Floating Conversion:* This conversion scheme transforms the parent solution into the required continuous solution.
  3. *DE Strategy:* The DE strategy transforms the parent solution into the child solution using its inbuilt crossover and mutation schemas.
  4. *Floating to Discrete Conversion:* This conversion schema transforms the continuous child solution into a discrete solution.
- **Mutation**
  5. *Relative Mutation Schema:* Formulates the child solution into the discrete solution of unique values.
- **Improvement Strategy**
  6. *Mutation:* Standard mutation is applied to obtain a better solution.
  7. *Insertion:* Uses a two-point cascade to obtain a better solution.
  8. *Repeat:* Execute steps 2-7 until reaching a specified cutoff limit on the total number of iterations.
- **Local Search**
  9. *Local Search:* Is initiated if stagnation occurs

Figure 1: EDE conceptual outline.

Population generation

The population for EDE is constructed using a random number generator. The solution is discrete and reflects the problem structure. This is unique to the incumbent process of having a differential population.

Discrete to floating and floating to discrete conversion

The approach for the conversion of discrete values into floating numbers and then back into discrete numbers after manipulation is accomplished through the utilization of Onwubolu's Approach (Price *et al.* 2006).

For the forward transformation from discrete to continuous numbers, the following formulation is used:

$$x_i' = -1 + x_i \cdot (1 - \varepsilon) \tag{3}$$

where ε is a small number. The values are transformed back into discrete numbers using:

$$x_i = \text{round}\left[(1 + x_i') \cdot (2 - \varepsilon)\right] \tag{4}$$

where the round function rounds the argument to the nearest integer.

DE Strategies

The most crucial and important factor in any heuristic is its internal manipulation routines. DE is highly effective due to its novel and robust internal mutation schemas (Price 1999). Price and Storn (2001) have described ten different working strategies of DE, which are usually dependent on the problem to be solved. Each strategy is dependent on three factors; the solution to be perturbed, number of different solutions considered for perturbation and the type of crossover used. The different strategies are given as:

(1) DE/best/1/(exp/bin): $u_i = x_{best} + F \cdot (x_{r1} - x_{r2})$

(2) DE/rand/1/(exp/bin): $u_i = x_{r1} + F \cdot (x_{r2} - x_{r3})$

(3) DE/rand-to-best/1/(exp/bin):
$$u_i = x_i + F \cdot (x_{best} - x_i) + F \cdot (x_{r1} - x_{r2})$$

(4) DE/best/2/(exp/bin):
$$u_i = x_{best} + F \cdot (x_{r1} - x_{r2} - x_{r3} - x_{r4})$$

(5) DE/rand/2/(exp/bin):
$$u_i = x_{r5} + F \cdot (x_{r1} - x_{r2} - x_{r3} - x_{r4})$$

The convention shown is of form DE/x/y/z, where DE stands for Differential Evolution, x represents the string denoting the solution to be perturbed, y is the number of different solutions to be perturbed and z is the type of crossover utilized. Two different types of crossover schemas are described; *binomial* (bin) and *exponential* (exp) crossover. Binomial crossover stipulates that crossover will occur on each of the *D* values in a solution whenever a randomly generated number between 0 and 1 is within the *CR* range. Exponential crossover is performed on the solution until the random value generated between 0 and 1 goes beyond the *CR* range.

Relative Mutation Schema

Since conversion is occuring between two operational domains, the number of infeasible solutions created will be significant. In order to have a larger number of valid solutions, it is imperative to have child repairing methods embedded. Three such methods are developed; *front* (FM), *back* (BM) and *random* mutation (RM).

Front mutation

FM utilizes the forward bias in its operation of changing the infeasible values in a solution. Starting from the first value and location one, the whole solution is scanned and the first occurrence of any value is regarded as feasible, while its second occurrence is regarded as infeasible.

$$u_{j,i} = \begin{cases} u_{j,i} & \text{if } u_{j,i} \notin \{u_{1,i},...,u_{j-1,i}\} \\ \tilde{u}_{j,i} & \text{if } u_{j,i} \notin \{u_{1,i},...,u_{j-1,i}\} \end{cases} \quad (5)$$

Whenever a infeasible soulton is detected, a random value is generated which is not in the solution and replaces the infeasible solution.

Back Mutation

BM is the direct opposite of FM, where the solution is scanned from the end, starting at the last value.

$$u_{j,i} = \begin{cases} u_{j,i} & \text{if } u_{j,i} \notin \{u_{j+1,i},...,u_{D,i}\} \\ \tilde{u}_{j,i} & \text{if } u_{j,i} \notin \{u_{j+1,i},...,u_{D,i}\} \end{cases} \quad (6)$$

Random Mutation

RM contains no bias for evaluation of the solution. A random array containing the indexes for the solution is created and this array is used to check the solution for repetition. Where ever a repetitive value is detected it is marked as infeasible.

Once the solution is checked for repetition, another array is created which contains the index of the infeasible solution. Using this array, the infeasible solution are replaced by feasible solutions using the random number generator.

$$A = \{y_1, y_2, y_3, ...y_D\}$$
$$\text{where } y_1 = \begin{cases} rnd[1, D] \\ \text{if } y_1 \notin \{y_1,...,y_{j-1}\} \end{cases} \quad (7)$$

$$u_{j,i} = \begin{cases} u_{y_j,i} & \text{if } u_{y_j,i} \notin \{u_{y_j,i},...,u_{y_{j-1},i}\} \\ \tilde{u}_{y_j,i} & \text{if } u_{y_{j,i}} \notin \{u_{y_j,i},...,u_{y_{j-1},i}\} \end{cases} \quad (8)$$

Improvement strategies

Improvement strategies are embedded into the heuristic in order to improve the solution. The two improvement strategies are *mutation* and *insertion*.

Mutation

Mutation is the movement of two individuals from a solution. This is done in order to find diversity in the solution. Two random numbers are generated and using them as index, the corresponding values in the solution are swapped. This solution is then evaluated for its fitness and if improvement is shown, then this new solution is accepted into the population.

Insertion

Insertion refers to the shift of the solution. A random number is generated and using this number as index, the two opposing sides of the solutions are swapped. This maintains the integrity of the solution and also allows the solution to possibly venture into diversified region of solution space.

Local search

Local search technique is used to find better solutions from the current solution utilizing some common mathematical techniques. In EDE, local search is only initiated when the population stagnates. Stagnation is idealized as non-improvement of the population over a period of five (5) generations. The local search technique accepted for this research is the 2-opt local search.

EXPERIMENT AND ANALYSIS

The experiment phase is divided into four segments. The first section discusses the different strategies and their effectiveness in solving permutative problems. The second section involves the testing of this approach over the DDE and GA. The third section compares the heuristics with constructive methods and the final sections does extensive evaluations with the Taillard benchmark problem sets.

Parameter Settings

The initial experimentation deals with the validation and selection of the control variables. There are three different variables in DE which are usd for fine tuning the heuristic; F, CR and DE Strategy. The following section were evaluated permutatively to find the optimal input values: CR = {0.1, 0.3, 0.5, 0.7, 0.9}, F = {0.1, 0.3, 0.5, 0.7, 0.9} and Strategy number = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. The different values were iteratively evaluated on the F15x25 data set. The lowest average value was produced by CR: 0.3 and F: 0.1. This was realized as the most stable parameter combination.

Using the above selected values, the second phase composed of selecting the best strategy. The results are presented in Table 1.

Table 1: Strategy selection

Strategy	Average
1	246.74
2	249.32
3	247.64
4	247.44
5	248
6	247.52
7	248.28
8	***245.8***
9	246.6
10	246.52

As observed, Strategy 8, on average performs better than the other strategies and was selected.

blu

Improvement over Discrete DE

The first section outlines the improvements on the generic discrete DE. The results are presented in Table 2.

Table 2: EDE comparison with generic DE

F XX	DE	GA	EDE	*% DE - GA	% EDE - DE	% EDE - GA
5 x 10	79.4	-	78	-	101.79	-
8 x 15	138.6	143	134	103.1	103.43	106.71
10 x 25	207.6	205	194	98.74	107.01	105.67
15 x 25	257.6	248	240	96.27	107.33	103.33
20 x 50	474.8	468	433	98.56	109.65	108.08
25 x 75	715.4	673	647	94.07	110.57	104.01
30 x 100	900.4	861	809	95.62	111.29	106.42
Ho-Chang	213	213	213	100	100.00	100.00

*% formulation: a-b $\%=\frac{a}{b}\times 100$

EDE has obtained better results than both GA and DE on the same problem instances. When comparing EDE to DE, column six shows that EDE outperforms DE, producing better results on each and every problem instance. For small sized problems the increase in small, for medium sized problems it is around 107%, while for large sized problems the improvement is in excess of 110%. These results validates that there has been a marked improvement from the previous DDE to the new EDE. The EDE has met the first objective in improving the DE.

In addition EDE to GA, which is widely considered as a benchmark optimization technique. Column seven of Table 9 shows that EDE outperforms GA on all the problem instances listed, from small sized problems to large problems. On average EDE is around 105% to GA results.

Comparison with Constructive Heuristics

The second section outlines the comparison of this approach with some established constructive heuristics. It is the general concensious that constructive heuristics are generally more robust sine they are targeted algorithms, where as metaheurists are generic algorithms.

Module two of the results are from the OR Library source, and are referenced in Ponnambalam *et al* (2001). These FSS problem instances are used by other researches and their finding have been published. These instances were evaluated in order to find the effectiveness of EDE compared to other algorithms inclusive of *constructive algorithms*.

The results are presented in Table 3.

Table 3: EDE comparison with Constructive Algorithms.

Instance	Size	Constructive Algorithm	GA	EDE	% to Optimal
Car 1	11 x 5	7038 (NEH)	**7036**	7038	99.97
Car 2	13 x 4	7410 (CDS)	**7160**	7166	99.91
Car 3	12 x 5	7399 (GUPT)	7489	**7312**	101.18
Car 4	14 x 4	8003 (NEH)	8003	**8003**	100.00
Car 5	10 x 6	8190 (NEH)	7748	**7720**	100.36
Car 6	8 x 9	9159 (NEH)	8501	**8397**	101.23
Car 7	7 x 7	6819 (CDS)	6590	**6590**	100.00
Car 8	8 x 8	8903 (CDS)	8366	**8366**	100.00
Hel 2	20 x 10	146 (NEH)	145	**139**	104.31
reC 01	20 x 5	1334 (NEH)	1350	**1249**	106.81
reC 03	20 x 5	1136 (NEH)	1189	**1111**	102.25
reC 05	20 x 5	1290 (PALM)	1307	**1249**	103.28
reC 07	20 x 10	1637 (NEH)	1700	**1584**	103.34
reC 09	20 x 10	1639 (CDS)	1616	**1574**	102.66
reC 11	20 x 10	1597 (CDS)	1550	**1464**	105.87
reC 13	20 x 15	2030 (NEH)	2120	**1957**	103.73
reC 15	20 x 15	2037 (NEH)	2115	**1984**	102.67
reC 17	20 x 15	2080 (RA)	2116	**1957**	106.28
reC 19	30 x 10	2189 (NEH)	2349	**2132**	102.38
reC 21	30 x 10	2157(NEH)	2262	**2065**	104.45
reC 23	30 x 10	2233(NEH)	2218	**2073**	106.99

NEH - Nawaz *et al* 1983; GUPT – Gupta 1971; PALM – Palmer 1965; CDS - Campbell *et al* 1970

A total of twenty-one problem instances were evaluated, with two different types of comparisons made. Out of the twenty-one problem instances, EDE obtained the optimal values for nineteen problem instances. For the other two problem instances it found results close to 99.9% to the optimal. On average EDE performed 101% to the optimal.

Comparison with Taillard Benchmark Problem Sets

The third experimentation module is referenced from Thaillard (1993). These sets pf problems have been extensively evaluated (see Nowicki *et al*. 1996 and Reeves *et al.* 1998). This benchmark set contains 100 particularly hard instances of 10 different sizes, selected from a large number of randomly generated problems.

A maximum of ten iterations was done for each problem instance. The population was kept at 100, and 100 generations were specified. The results represented in Table 4 are as quality solutions with the percentage relative increase in makespan with respect to the upper bound provided by Thaillard (1993). To be specific the formulation is given as:

$$\Delta_{avg} = \frac{(H - U) \times 100}{U} \qquad (8)$$

where H denotes the value of the makespan that is produced by the EDE algorithm and U is the upper bound or the lower bound as computed.

The results obtained are compared with those produced by GA, Particle Swarm Optimization (PSO_{spv}) DE (DE_{spv}) and DE with local search ($DE_{spv+exchange}$) as in Tasgetiren *et al.* (2004). The results are tabulated in Table 4.

Table 4: EDE comparison with DE spv and PSO over the Taillard benchmark problem sets.

	GA		**PSO_{spv}**		**DE_{spv}**		**$DE_{spv+exchange}$**		**EDE**	
	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}
20x5	3.13	1.86	1.71	1.25	2.25	1.37	0.69	0.64	0.98	0.66
20x10	5.42	1.72	3.28	1.19	3.71	1.24	2.01	0.93	1.81	0.77
20x20	4.22	1.31	2.84	1.15	3.03	0.98	1.85	0.87	1.75	0.57
50x5	1.69	0.79	1.15	0.70	0.88	0.52	0.41	0.37	0.40	0.36
50x10	5.61	1.41	4.83	1.16	4.12	1.10	2.41	0.90	3.18	0.94
50x20	6.95	1.09	6.68	1.35	5.56	1.22	3.59	0.78	4.05	0.65
100x5	0.81	0.39	0.59	0.34	0.44	0.29	0.21	0.21	0.41	0.29
100x10	3.12	0.95	3.26	1.04	2.28	0.75	1.41	0.57	1.46	0.36
100x20	6.32	0.89	7.19	0.99	6.78	1.12	3.11	0.55	3.61	0.36
200x10	2.08	0.45	2.47	0.71	1.88	0.69	1.06	0.35	0.95	0.18

Through the analysis of Table 4, it can be observed that EDE compares outstandingly with other algorithms. EDE basically outperforms GA, PSO and $DE_{spv.}$ The only serious competition comes from the new variant of $DE_{spv+exchange.}$ EDE and $DE_{spv+exchange}$ are highly compatible. EDE outperforms $DE_{spv+exchange}$ on the data sets of 20x10, 20x20, 50x5 and 200x5. In the remainder of the sets EDE performs remarkbly to the values reported by $DE_{spv+exchange}$. On average EDE displays better standard deviation than that of $DE_{spv+exchange}$. This validates the consistancy of EDE compared to $DE_{spv+exchange}$.

CONCLUSION

The new enhanced variant of differential evolution (EDE) algorithm has been proposed and found effective in solving a range of difficult flow shop scheduling problems. The different experimentations have validated the effectiveness of EDE.

EDE has shown marked impovement over the DDE approach, and has performed outstandingly against the constructive algorithms. The final validation has been done by extensive evaluation with Taillard problem sets and has been found to perform comparatively with other new emerging algorithms such as GA, PSO_{spv}, DE_{spv} and $DE_{spv+exchange.}$

EDE is shown as a versatile and robust new algorithm which has improved and enhanced the basic principles of DE. The new enhancement routines that have been embedded into DE have proven effective in enhancing the performance of DE in the scheduling problem of Flow Shop.

REFERENCES

Campbell, H. G., Durek, R.A., and Smith, M.L. 1970. "A heuristic algorithm for the *n* job *m* machine sequencing problem", *Management Science,* 16, (B) 630-637.

Goldberg, D., E. 1989. "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Publishing Company, Inc.

Lampinen, J. and Storn, R. 2004. "Differential Evolution", In: *New Optimization Techniques in Engineering*, Onwubolu, G., Babu, B., (Eds.), Springer Verlag, Germany, 123-163.

Lampinen, J. and Zelinka, I. 1999. "Mechanical engineering design optimization by Differential evolution", In *New Ideas in Optimization*, : Corne, D., Dorigo, M. and Glover (Eds.), McGraw Hill International, UK, 127-146.

Nawaz, M., Enscore, E. E. Jr, and Ham, I., 1983. "A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem", *OMEGA International Journal of Management Science* 11, 91-95.

Nowicki, E., Smutnicki, C. 1996. "A fast tabu search algorithm for the permutative flow shop problem". *European Journal of Operations Research*, 91, 160-175

Onwubolu, G. C. and Mutingi, M. 1999. "Genetic algorithm for minimizing tardiness in flow-shop scheduling", *Production Planning and Control* 10 (5) 462-471.

Onwubolu, G. C. 2001. "Optimization using Differential Evolution Algorithm", *Technical Report TR-2001-05*, IAS, October 2001.

Onwubolu G. C. 2002. *Emerging Optimization Techniques in Production Planning and Control*. Imperial Collage Press, London.

Onwubolu, G. C. 2004. "Optimizing CNC Drilling Machine Operations: Traveling Salesman Problem-Differential Evolution Approach". In: *New Optimization Techniques in Engineering*, Onwubolu, G., Babu, B., (Eds.), Springer Verlag, Germany, 537-564.

Onwubolu, G. C. and Davendra, D. 2006. "Scheduling flow shops using differential evolution algorithm". *European Journal of Operations Research*. 171, 674-679.

OR Library: http://mscmga.ms.ic.ac.uk/info.html

Pinedo, M. 1995. *Scheduling: theory, algorithms and systems*, Prentice Hall, Inc., New Jersey.

Ponnambalam, S. G., Aravindan, P. and Chandrasekhar, S. 2001. "Constructive and improvement flow shop scheduling heuristic: an extensive evaluation", *Production Planning and Control* 12**,** 335-344.

Price, K. 1999. "An introduction to differential evolution". In: *New Ideas in Optimization*, Corne, D., Dorigo, M., and Glover (Eds.), McGraw Hill International, UK, 79-108.

Price, K,. Storn, R. and Lampinen, R. 2006. *Differential Evolution: A practical approach to global optimization.* Springer Verlag, Germany.

Reeves, C. and Yamada, T. 1998. "Genetic Algorithms, path relinking and flowshop sequencing problem". *Evolutionary Computation* 6, 45-60.

Taillard, E. 1993. "Benchmarks for basic scheduling problems". *European Journal of Operations Research*, 64, 278-285.

Tasgetiren, M. F., Sevkli, M. Liang, Y-C., and Gencyilmaz, G. 2004. "Particle swamp optimization algorithm for permutative flowshops sequencing problems", *4th International Workshops on Ant Algorithms and Swamp Intelligence*, ANTS2004, LNCS 3127, Brussel, Belgium. (Sept) 5-8, 389-390.

Tasgetiren, M. F., Liang, Y-C., Sevkli, M. and Gencyilmaz, G. 2004. "Differential Evolution Algorithm for Permutative Flowshops Sequencing Problem with Makespan Criterion*:, 4th International Symposium on Intelligent Manufacturing Systems*, IMS2004, Sakaraya, Turkey. (Sept) 5-8, 442-452.

DONALD D. DAVENDRA is a Master of Science graduate in Optimization Techniques from the University of the South Pacific, Fiji Islands. Currently he is a doctoral candidate in Technical Cyberrnatics at the Tomas Bata University in Zlin, Czech Republic. His email address is davendra@fai.utb.cz

GODFREY C. ONWUBOLU is the Professor, Chair and Head of Engineering at the University of South Pacific. He is the author of three books in optimization techniques and mechatronics. His research interests are in optimization, intelligent manufacturing and mechatronics. His email address onwubolu_g@usp.ac.fj