Zuzana Čičková
Stanislav Števo

Flow Shop Scheduling using Differential Evolution

Article Info:

Management Information Systems, Vol. 5 (2010), No. 2, pp. 008-013

Received 14 January 2010
Accepted 2 August 2010

UDC 005.311.12:517.9 ; 658.6 ; 007:005]:004

Summary

The article is focused on the application of differential evolution for solving flow shop problem that belongs to the class of scheduling problems. The scheduling problems arise in diverse areas such as manufacturing systems, production planning, computer design, logistics etc.. Only in very special cases there exist exact polynomial algorithms to reach optimal solution. In most of the other cases, its computational complexity is NP-hard and it seems to be desirable to employ some heuristics to solve it. Nowadays, the use of some methods that are based on metaheuristics is a popular way. One of them is a differential evolution, which belongs to the class of evolutionary techniques. The application of evolutionary algorithms to NP-hard problems generally requires a special modification of these algorithms; therefore the main object of the work is to adapt a canonical version of differential evolution for solving flow shop problem. The effectiveness of the proposed approach is compared with other evolutionary techniques known from the already published results. The available instance of flow shop Car and Rec are used for comparison.

Key words

Flow Shop Problem, Scheduling, Differential Evolution

Introduction

Flow shop systems are known in the field of production logistics which is called scheduling theory. This theory includes complicated schedules e.g. production schedules and school schedules, transportation, personal and many others (Palúch & Peško, 2006).

Next, we will introduce some basic concepts of production scheduling. The basic elements are:

- *operation – o*, basic technological operation, which is not divisible to the partial operations,
- *job – J*, is a set of operations $\{o_1, o_2, ..., o_n\}$, which can not be done in the same time,
- *machine – M*, a device capable of performing one or several types of operations.

Flow shop system can be basically characterized as systems in which the operations $\{o_1, o_2, ..., o_n\}$ (individual operations are uninterruptiblne) of every job J must be processed on machines $\{M_1, M_2, ..., M_m\}$ (machines are always available and they can process only one operation at time) in the same order. There are no precedence constraints among operations of different jobs and also the processing time p_{ij}, $i = 1,2, ...n, j = 1,2, ...m$, for every operation o_i, $i = 1,2, ...n$, on each machine M_j, $j = 1,2, ...m$, is known. The problem is to find the job sequences on the machines which minimize the makespan, i.e. the maximum of the completion times of all operations. Makespan is the most frequently used criterion for flow shop and it means to minimize the end time of final operation on the last machine (Makespan - C_{max}) so the whole processing time (objectives as to minimize mean flowtime, total terdiness etc.. are also possible). Other basic assumptions of scheduling can be found in e.g. (Palúch & Peško, 2006).

Flow shop systems with two machines, where the aim is to minimize C_{max}, can be solved by a polynomial *Johnsons algorithm* (full description in (Palúch & Peško, 2006) (Brezina et al., 2009)), but there is no polynomial algorithm for solving so that problems with three and more machines (except the special cases such as those described in (Palúch & Peško, 2006)). Generally, flow shop problem is NP- hard with number of possible schedulling $(n!)^{(m-1)}$. There are various ways to classify algorithms for solving so that problem, each with its own merits. One way to classify algorithms is by implementation principle (Čičková et al., 2008):

Explicit enumeration. It leads to reconnaissance all possible solutions of problem, therefore is applicable only for problem of small size.

Deterministic methods. These algorithms base only on rigorous methods of „classical" mathematics. Some additional information, such as gradient, convexity etc. is usually needed (branch and bound algorithm, cutting plane method, dynamic programming etc.).

Stochastic methods. These algorithms work on probabilistic methods to solve problems. Stochastic algorithms work slowly and are applicable only for

„guessing“ (monte carlo, random search walk, evolutionary computation etc.).

Combined methods. Combined methods are comprised by stochastic and deterministic composition. Various metaheuristics algorithm has been devised (ant colony optimization, memetic algorithms, genetic algorithms etc.). Metaheuristics consist of general search procedures whose principles allow them to escape local optimality using heuristics design. Evolutionary algorithms are significant part of metaheuristics.

1. Differential Evolution

Differential evolution belongs to the class of evolutionary techniques, where the best known representatives are genetic algorithms, but there are some differences e.g. an individual is created with the use of four parents and it is mutedet two times etc.. The evolutionary algorithms comprise a large number of nontraditional computing techniques whose common characteristic is that they are inspired by the observation of the nature processes (genetic algorithms, ant colony optimization, differential evolution, etc.), respectively from other disciplines (e.g. simulated annealing). Nowadays evolutionary algorithms are considered to be effective tools that can be used to search for solutions of optimization problems. The big advantage over traditional methods is that they are designed to find global extremes (with built-in stochastic component) and that their use does not require a priori knowledge of optimized function (convexity, differential etc.).

Evolutionary algorithms differ from more traditional optimization techniques in that they involve a search from a "population" of individuals, not from a single one. Each individual represents one candidate solution for the given problem that is represented by parameters of individual. Associated with each individual is also the *fitness*, which represents the relevant value of objective function. A population can be viewed as *np x d* matrix (*np* - number of individuals in the population, *d* – number of parameter of individual). Every step involves a competitive selection that is carried out poor solutions.

Differential evolution was introduced by Price and Storm (Storn & Price, 1997). Differential evolution, as well as other evolutionary techniques, works well on solving non-constrained problems that contain continuous variables, but nowadays there were developed few approaches that involve the solving of constrained problems with integer or binary variables. Varied approaches were used with more or less success for solving many real problems etc. traveling salesman problem, vehicle routing problem etc. (see (Brezina et al., 2009), (Onwubolu & Babu, 2004), (Palúch & Peško, 2006)).

The principle of basic verzion of differential evolution can be described by following pseudocode:

```
BEGIN
SETTING of control parameters;
INITALIZATION of population;
    EVALUATION of each individual;
    WHILE (STOPPING CRITERION is not satisfied) DO
    FOR (each individual of the population) DO
(REPRODUCTIVE CYCLE):
CREATE differential vector
CREATE trial vector
CREATE test vector
IF (EVALUATION of test vector)> (EVALUATION of current selected individual)
THEN (SUBSTITUDE the selected individual with the test vector)
ENDIF
    ENDFOR
    ENDWHILE
    EVALUATE process of calculating
END
```

The steps of the algorithm can be briefly summarized as follows (according to (Zelinka, 2002) (Onwubolu & Babu, 2004), where it is possible to found recommended values for different parameters):

Setting of the control parameters. Differential evolution is controlled by a special set of parameters. Recommended values for the parameters are usually derived empirically from experiments:

d – dimensionality. Number of parameters of individual (usually also number of arguments of objective function).

np – population size. Number of individuals in population. recommended setting is 5d to 30d, respectively 100d, in cace the optimized funcion is multimodal (Zelinka, 2002).

g – generations. Represent the maximum number of iteration (*g* is also stopping criterion).

cr – crossover constant, $cr \in \langle 0,1 \rangle$. If the *cr* value is set to 0, the mutation test vector is only a copy of the current (fourth) parent. If the *cr* value is set to 1, test vector will be created only by three randomly selected parents and a differential evolution algorithm is purely stochastic.

f– mutation constant, $f \in \langle 0,1 \rangle$. *f* specifies the level of stochastics in the evolutionary process.

Initialization. The population must be initialized at the beginning of evolutionary process. Usually the random initialization is used so that each individual represents a candidate solution of given problem, respectively it can be also used some information about the problem (if available). Each individual is then evaluated with the *fitness* (relevant value of objective function).

The test of stopping condition. In its canonical form, the only stopping criterion is to reach the maximal number of iterations (represent by parameter *g*). The user may change program with some different type of ending the parameter, eg. stop evolution or reformulate the control parameters, if over th over the last fixed number of generations has not changed the value of best individual and so on.

Reproductive cycle. This cycle comprise the crossing and mutation to create individuals for the next generation. For each individual $\mathbf{x}_i^g$, *i* =1,2, ...*np,* from the population another three different individuals are chosen (vectors r1, r2, r3). The difference of the first two vectors (r1 and r2) gives the *differential vector* **d**, which is multiplied by mutation constant *f* and added to vector r3. Thus, we get *trial vector* **v**. Formally:

$$\mathbf{v}_j^t = x_{r3,j}^t + f * \left(x_{r1,j}^t - x_{r2,j}^t \right)$$
$$j = 1,2...,d \text{ , } t = 1,2...,g \qquad (1)$$

After the mutation process comes the formation of a new individual, which is also called *test vector* $\mathbf{x}^{\mathbf{test}}$ so that one element after another is selected from the currently selected individual $\mathbf{x}_i^g$ and from the trial vector **v** and for every pair is generated a random number from the interval <0.1>, which is compared with the crossing constant *cr*. If the generated random number is less than or equal to *cr*, to the relevant position of $\mathbf{x}^{\mathbf{test}}$ comes the element of *trial vector* **v,** otherwise of current selected individual $\mathbf{x}_i^g$**.**

Formally:

$$x_j^{test} = \begin{cases} x_{r3j}{}^g + f\left(x_{r1j}{}^g - x_{r2j}{}^g\right), \text{ if } rand_j \langle 0,1 \rangle \le cr \ \vee \ j = k \\ \\ x_{ij}{}^g, \text{ otherwise} \end{cases} \qquad (2)$$

where

$$i = 1,2,...,np \text{ , } j = 1,2,...,d \text{ , } k \in \{1,2,...,d\} \qquad (3)$$
$$r1, r2, r3 \in \{1,2,...np\} \text{ ; } r1 \ne r2 \ne r3 \ne i$$

where *k* is a random index, which always ensures a change of at least one parameter in the test vector. The value of the objective function for the test vector is compared to the value of objective function of the curent selected individual and to the next generation is selected the vector with the better objective value.

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{x}^{test}, \text{ ak} f_{\cos t}(\mathbf{x}^{\text{test}}) \le f_{\cos t}(\mathbf{x}_i^g) \\ \\ \mathbf{x}_i^g, \text{ otherwise} \end{cases} \qquad (3)$$

So that process continues in each generation for all individuals. The result is a new generation with the same number of individuals.

Evaluation. The whole process of reproduction continues until the last (users specified) number of generations is reached. The value of the best individual from each generation is reflected to *history vector*, which shows the progression of an evolutionary process.

2. Flow shop problem using differential evolution

Since the differential evolution is an algorithm, which works well in the case of non-constrained problems with continuous variables, in applying the algorithm for solving NP-hard problems, is necessary to consider the following factors:

- Selection of an appropriate representation of individual
- Formulation of objective function
- Transformation the parameters of individual to the real numbers
- Transformation of unfeasible solutions
- Setting of the control parameters of the differential evolution

Selection of an appropriate representation of individual. A natural representation of individual, that is particularly known from genetic algorithms (where has been often used with success by solving much known traveling salesman problem), was chosen. In response to the flow shop problem, each operation is assigned with integer from 1 to *n* (*n* represents the number of operations), which represents corresponding operation in individual. Each individual is then represented by a *d*-dimensional vector of integers, representing direct the order of processing operations. Then, the initial population is generated as follows:

$$\mathrm{P}^{(0)} = x_{i,j}^{(0)} = \text{randperm}\left(d\right) \quad i = 1,2,...,np \;\; j = 1,2,...,d \qquad (4)$$

where the function randperm (*d*) ensure the establishment of a random permutation of *d* integers, so it is the random permutation of the sequence of operations. Each individual in the population is also assigned with its fitness that represents maximum of the completion times of all operations *(Makespan)*.

Formulation of objective function. The computation of objective function value for an individual is carried out in two steps with respect the following facts:

1. The first machine begins to process the first operation in sequence in time 0 and this operation will not proceed on the next machine unless its processing on first machine is done.
2. Processing of other operations in the sequence is as follows:

- the first machine works without downtime, i.e. operations are assigned in the order, which is determined by individual
- the processing of relevant operation on the further machines is conditioned by fact that the machine is already free (if not, relevant operation waits for completion of previous operation on this machine)
- and also the processing of relevant operation tasks on the previous machine is done (if no, there is a machine downtime, waiting for the operation).

Transformation the parameters of individual to the real numbers. Because the differential evolution algorithm was originally designed to solve problems with continuous-time variables and the used natural representation consists of integer variables, it is desirable to transform integers to real numbers. The used method for transformation was presented in (Onwubolu & Babu, 2004) for solving traveling salesman problem. Let z_i, $i = 1, 2, ..., n$ represents an integer number. The equivalent continuous variable for z_i is given as:

$$r_i = -1 + \frac{z_i * f * 5}{10^3 - 1} \qquad (5)$$

where f is given, for example. f = 200.

Reverse transformation of real numbers to integers (used to evaluate the objective function):

$$z_i = \frac{(1 + r_i) * (10^3 - 1)}{5 * f} \qquad (6)$$

$$\alpha = \mathrm{int}(z_i + 0,5) \qquad (7)$$

$$\beta = \alpha - z_i \qquad (8)$$

Transformation of unfeasible solutions. The use of differential evolution algorithm does not require the formation of feasible solution in case of natural representation of individual, therefore it is necessary to choose an appropriate method of transformation of the unfeasible solutions. The problem of infeasibility occurs in two cases:

a) Parameter of individual after the transformation from real numbers to integers is less then 1 or greater then *d*, in this case the relevant parameter is replaced by new randomly generated parameters in range 1 to *d*.
b) Created individual does not comprise a permutation of integers *d*. In this case, the correction approach that was presented in (Brezina et al, 2009) by solving vehicle routing problem, was used.

1) Let **m** is the vector of parameters of the individual dimension of *d* with *k* different elements. If *d* - *k* = 0, go to step 4). Otherwise, go to step 2).
2) Create the vector **p** (dimension *d* - *k*) of random permutation of such *d* – *k* elements, which are not included in the vector **m**. If the number of non-zero components of the vector **p** = 0, go to step 4). Otherwise, find the first repeated element of vector **m**. Let this element be m_c and let the first nonzero element of vector **p** be p_k . Set $m_c = p_k$ and go to step 3).
3) Set pk = 0 and return to the step 2).
4) Return **m**

As part of the software support for experimens, the MATLAB 7.1 was used. Two functions were created: the differential evolution algorithm adapted for solving flow shop systems and the function for objective function calculation that calculates the value of optimized function.

3. Experiments

To discover the effectiveness of the presented techniques, the free available test data *Car1* up to *Car8* from OR library were used. [1] This Flow shop instances were investigated by many researchers who applied a variety of techniques to solve it (and also there is a known optimal value of objective function). Firstly, the 165 simulation were carried out to determine the effective setting of control

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/ (15.9.2010)

parameters *f* and *cr* (with the simultanous use of the set *np* = 300, *g* = 1500). The best parameters obtained from combining these parameters during experimentation are: *cr* = 0,1 and *f* = 0,05. Altroght those setting presents a relative low crossing rate and mutation, they seemed to be adequate to garantee evolution process. After all simulation we can resume that we were able to achive optimal solution for all instances *Car*.

Further on, the test data Rec 03 up to Rec 17 also from OR library were solved with the use of control parameters: *cr* = 0,1, *f* = 0,05, *np* = 300, *g* = 4500 and the test data Rec 19 up to Rec 29 with the use of control parameters: *cr* = 0,1, *f* = 0,05, *np* = 500, *g* = 8500. For each problem three simulations were done. The results are given in Table 1:

Table 1 Results for Rec instances[2]

Problem	Size	Simulation1	Simulation2	Simulation3	Mean	Minimum
Rec03	20x5	1111	1111	1110	1110,67	1110
Rec05	20x5	1245	1245	1245	1245	1245
Rec07	20x10	1566	1566	1566	1566	1566
Rec09	20x10	1537	1537	1537	1537	1537
Rec11	20x10	1431	1431	1431	1431	1431
Rec13	20x15	1935	1936	1935	1935,33	1935
Rec15	20x15	1964	1961	1962	1962,33	1961
Rec17	20x15	1921	1919	1909	1916,33	1909
Rec19	30x10	2106	2109	2112	2109	2106
Rec21	30x10	2046	2046	2046	2046	2046
Rec23	30x10	2032	2032	2032	2032	2032
Rec25	30x15	2559	2599	2553	2570,33	2553
Rec27	30x15	2412	2399	2410	2407	2399
Rec29	30x15	2336	2325	2333	2331,33	2325

In order to compare the before mentioned approach with other minimizing strategies the work (Davendra & Zelinka, 2008) was used. There were published the results of following five evolutionary algorithms:

H-GA – Hybrid Genetic Algorithm
OGA – Othogonal Genetic Algorithm
IGA – Improved Genetic Algorithm
MAEA – Multiagent Evolutionary Algorithm
SOMA – Self Organizing Migrating Algorithm

The compatrision of differential evolution with these evolutionary approaches is given in Table 2, where the percentage deviations from optimal value for different evolutionary techniques are seen.

Table 2 Comparison with other heuristics

Proble m	Size	Opti mum	H-GA	OG A	IGA	MAE A	SOM A	DE
Car1	11x5	7038	0	0	0	0	0	0
Car2	13x4	7166	0	0	0	0	0	0
Car3	12x5	7312	0	0	0	0	0	0
Car4	14x4	8003	0	0	0	0	0	0
Car5	10x6	7720	0	0	0	0	0	0
Car6	8x9	8505	0	0	0	0	0	0
Car7	7x7	6590	0	0	0	0	0	0
Car8	8x8	8366	0	0	0	0	0	0
Rec01	20x5	1247	0	0,04	0	0	0	0
Rec03	20x5	1109	0	0	0	0	0	0,18
Rec05	20x5	1242	0,08	0,21	0	0	0,002	0,2
Rec07	20x10	1566	0	0,79	0	0	0,01	0
Rec09	20x10	1537	0	0,35	0	0	0	0
Rec11	20x10	1431	0	0,91	0	0	0	0
Rec13	20x15	1930	0,52	1,08	0,62	0	0	0,21
Rec15	20x15	1950	0,92	1,23	0,46	0	0,01	0,56
Rec17	20x15	1902	1,26	2,08	1,73	0	0,02	0,37
Rec19	30x10	2093	0,38	1,76	1,09	0,28	0,02	0,91
Rec21	30x10	2017	0,89	1,64	1,44	0,44	0,02	1,44
Rec23	30x10	2011	0,45	1,9	0,45	0,44	0,03	1,04
Rec25	30x15	2513	1,03	2,67	1,63	0,43	0,03	1,59
Rec27	30x15	2373	1,18	2,09	0,8	0,56	0,01	1,56

From the last column of Table 2 it is evident that the percentage deviation of results obtained by differential evolution algorithm from the optimal solution was less than 1.7%. The results for the instance *Car* are consistent with the optimal solution in every case. Problems Rec01 up to Rec17 were solved with the percentage deviation from optimal value within the range from 0 to 0.56% and problems Rec19 up to Rec29 within the range from 0.91 to 1.67%.

Based on these results it can be stated that the differential evolution presents a powerfull approach for solving flow shop problems. To deal with the number of operations less than 20, the optimal solution was obtained in every case, in the case of sequencing of 20 operacions, the percentage deviations from the optimal solution were less than 1% and in case of sequencing of 30 operations the percentage deviations from the optimal solution were always less than 1.7%. As it is seen from the Table 2, presented approach is fully comparable with the published results of other evolutionary techniques.

[2] First number represents the number of operations, second number represents the number of machines.

Conclusion

Flow shop problem is one of NP-hard problems. A typical conflict in dealing with this kind of problems arises between the time available for the calculation and the quality of the solution. While the exact methods (eg, branch and bound algorithm) are often able to identify the optimum in unacceptably long time, the quality of solutions obtained by heuristic may be disputable. Nowadays, we follow the increased interest in methods, which are inspired by different biological evolutionary processes in nature. This technology is covered by the common name of "evolutionary algorithms". But their application to constrained problems requires some additional modifications of theirs basic versions. The paper was focused on application of differential evolution to flow shop problem. The special factors that involve the use of differential evolution were presented and the efficiency of calculations has been validated on the basis of publicly available instances. Based on presented results it can be concluded that the proposed approach is quite powerful in dealing with flow shop problem and it is fully comparable with the published results of other evolutionary techniques.

References

Brezina, I., Čičková, Z., & Reiff, M. (2009). *Kvantitatívne metódy na podporu logistických procesov.* Bratislava: Ekonóm.

Brezina, I., Čičková, Z., Gežik, P., & Pekár, J. (2009). *Modelovanie reverznej logistiky – optimalizácia procesov recyklácie a likvidácie odpadu.* Bratislava: Ekonóm.

Čičková, Z., Brezina, I., & Pekár, P. (2008). Alternative method for solving traveling salesman problem by evolutionary algorithm. *Management information systems , 3* (1), 17-22.

Davendra, D., & Zelinka, I. (2008). Flow Shop Scheduling using Self Organizing migration Algorithm. *22nd European Conference on Modelling and Simulation*, (pp. 195-200). Nicosia.

Onwubolu, G., & Babu, B. (2004). *New Optimization Techniques in Engineering, Studies in Fuzziness and Soft Computing.* New York: Springer.

Paluch, S., & Peško, Š. (2006). *Kvantitatívne metódy v logistike.* Žilina: EDIS vydavateľstvo ŽU.

Sekaj, I. (2005). *Evolučné výpočty a ich využitie v praxi.* Bratislava: IRIS.

Storn, R., & Price, K. (1997). Differential Evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Jounal of Global Optimization , 11* (4), 341-359.

Zelinka, I. (2002). *Umělá inteligence v problémech globální optimalizace.* Praha: BEN-technická literatúra.

Zuzana Čičková
University of Economics in Bratislava
Department of Operations Research and Econometrics, Faculty of Economic Informatics
Dolnozemská cesta 1/b
852 35 Bratislava
Slovak Republic
Email: cickova@euba.sk

Stanislav Števo
Slovak University of Technology
Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology
Ilkovičova 3
852 35 Bratislava
Slovak Republic
Email: stanislav.stevo@stuba.sk