Solution of Flow Shop Scheduling Problems using the Differential Evolution Algorithm

Fran Sérgio Lobato¹, Rubens Gedraite², Sergio M. S. Neiro³

¹²³Laboratory of Modeling, Simulation, Control and Optimization of Process, School of Chemical Engineering, Federal University of Uberlândia, Av. João Naves de Ávila, 2121, Uberlândia, MG, ZIP CODE 38400-902, Brazil
¹fslobato@feq.ufu.br, ²gedraite@feq.ufu.br, ³sergioneiro@feq.ufu.br

1. Abstract
During the past two decades, there have been increasing interests on permutation flow shop with different types of objective functions such as minimizing the makespan, the weighted mean flow-time etc. The permutation flow shop is formulated as a mixed integer programming and it is classified as NP-Hard problem. This problem is normally classified as a complex combinatorial optimization problem, in which there is a set of jobs to be processed in a set of machines in the same order. We normally look for a special sequence of processing the jobs in the machines to minimize one or more criteria such as minimization of makespan, mean flow, etc. In this contribution, the Differential Evolution algorithm is used to minimize the makespan for the permutation flow shop scheduling problem. The results of the proposed method are compared with a classical strategy implemented in GAMS (General Algebraic Modeling System) software. The results obtained indicate that the proposed methodology characterizes a promising alternative for dealing with this type of problem.

2. Keywords: Permutation Flow Shop, Differential Evolution Algorithm, Mixed Integer Programming.

3. Introduction
In the last decades, the flow shop scheduling problem (FSSP) with makespan minimization has been a central and well-studied problem in combinatorial optimization community, known by its intractability form theoretical and computational aspects [1]. This problem is classified as a complex combinatorial optimization problem, in which there is a set of n jobs (1, …, n) to be processed in a set of m machines (1, …, m) in the same order [2]. We normally look for a special sequence of processing the jobs in the machines to minimize one or more criteria such as minimization of makespan, mean flow, among others. There are different most commonly used criteria such as the minimization of the total completion time or makespan of the schedule (Cmax) which is sometimes referred to as maximum flow time. The processing times needed for the jobs on the machines are assumed to be non-negative and deterministic denoted as pij with i=1, …, n and j=1, …, m [2].

Recently, algorithms based on heuristic and meta-heuristics approaches have been used to solve the FSSP due to the success observed in the solution of general optimization problems. Turner and Booth [3] compared two famous heuristics with a set of 350 random problems. Ruiz and Maroto [4] presented a review and comparative evaluation of heuristics and meta-heuristics for the permutation flow shop problem with the makespan criterion. Lian et al. [5] applied an efficient similar particle swarm optimization algorithm to the PFSS problem with the objective of minimizing the makespan. Tasgetiren et al. [6] solved the permutation flow shop sequencing problem (PFSP) with a particle swarm optimization algorithm considering the objectives of minimizing makespan and the total flow time of jobs. Naderi and Ruiz [7] studied a new generalization of the regular permutation flow shop scheduling problem referred to as the distributed permutation flow shop scheduling problem. Under this generalization, they assumed that there are a total of H identical factories or shops, each one with m machines disposed in series. A set of n available jobs have to be distributed among the H factories and then a processing sequence has to be derived for the jobs assigned to each factory. Their optimization criterion was the minimization of the maximum completion time or makespan among the factories. Dong et al. [8] presented an integrated local search algorithm to solve the permutation flow shop sequencing problem with total flow time criterion. They showed the effectiveness and superiority of their method over three constructive heuristics, three ant-colony algorithms and a particle swarm optimization algorithm. Vallada and Ruiz [9] presented three genetic algorithms for the permutation flow shop scheduling problem with total tardiness minimization criterion. The algorithms include advanced techniques like path re-linking, local search and a procedure to control the diversity of the population. Koh et al. [10] proposed several heuristics including one genetic algorithm for three different objective functions: total completion time, total weighted completion time and makespan. In addition, the simulated annealing algorithm has been successfully applied to solve a variety of scheduling problems that include project scheduling [11,12], parallel machines [13,14], and flow shops [15,16].

In the present contribution the DE is used to solve a permutation flow shop scheduling problem with the objective of minimizing the makespan. The results obtained with this methodology are compared with others strategies. This work is organized as follows. The mathematical formulation of permutation flow shop problem is presented in section 4. A review of the DE is presented in section 5. The results and discussion are described in Section 6. Finally, the conclusions and suggestions for future work conclude the paper.

4. Mathematical Model for Permutation Flow Shop Scheduling Problem
As pointed out in the previous sections, the aim of the present work is to test the efficiency of the DE algorithm in solving scheduling problems by varying a number of the DE algorithm parameters and analyzing the corresponding results. For this purpose, an example problem that has been extensively discussed in the literature was picked as a case study. Products 1 and 2 are to be produced from
three different feedstocks A, B and C. Figure 1 shows the state task network (STN) representation of the problem involving five tasks and four intermediate materials.

The recipe is as follows:

1. **Heating**: feedstock A is heated for 1 h by the heating task on the heater unit;
2. **Reaction 1**: reaction 1 takes 50% of feedstock B and 50% of feedstock C and let them react for 2 h to form intermediate BC. Two pieces of equipments are able to perform reaction 1: reactors 1 and 2;
3. **Reaction 2**: reaction 2 takes 40% of Hot A and 60% of intermediate C and let them react for 2 h to form product 1 (40%) and intermediate AB (60%). Two pieces of equipments are able to perform reaction 2: reactors 1 and 2;
4. **Reaction 3**: reaction 3 takes 20% of feedstock C and 80% of intermediate AB and let them react for 1 h to form impure E. Two pieces of equipments are able to perform reaction 3: reactors 1 and 2;
5. **Separation**: impure E is distilled on the still to separate pure product 2 (90%) after 1 h and pure intermediate AB (10%) after 2 h. Intermediate AB is recycled.

Available equipment capacity:
- Heater: 100 kg
- Reactor 1: 80 kg
- Reactor 2: 50 kg
- Still: 200 kg

Available storage capacity:
- Feedstock A, B and C: unlimited
- Hot A: 100 kg
- Intermediate AB: 200 kg
- Intermediate BC: 150 kg
- Impure E: 100 kg
- Products 1 and 2: unlimited

The mathematical formulation as originally proposed by Kondili et al. [17] is presented in this section according to the nomenclature that follows:

<table>
<thead>
<tr>
<th>Indices</th>
<th>Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>set of all tasks</td>
</tr>
<tr>
<td>( J )</td>
<td>Tasks</td>
</tr>
<tr>
<td>( S )</td>
<td>Units</td>
</tr>
<tr>
<td>( T )</td>
<td>States</td>
</tr>
<tr>
<td></td>
<td>time periods</td>
</tr>
</tbody>
</table>

### Figure 1. STN representation of the problem involving five tasks and four intermediate materials [17].
\begin{equation}
J \quad \text{set of all units}
\end{equation}
\begin{equation}
S \quad \text{set of all states}
\end{equation}
\begin{equation}
T \quad \text{set of all time periods}
\end{equation}
\begin{equation}
S_R \quad \text{set of states corresponding to raw materials}
\end{equation}
\begin{equation}
S_P \quad \text{set of states corresponding to final products}
\end{equation}
\begin{equation}
I_j \quad \text{set of processing tasks that can be performed by unit } j
\end{equation}
\begin{equation}
K_i \quad \text{set of units that are able to perform task } i
\end{equation}
\begin{equation}
T_s^p \quad \text{set of tasks which produce material at state } s
\end{equation}
\begin{equation}
T_s^c \quad \text{set of tasks which consume material from state } s
\end{equation}

### Parameters

\begin{description}
\item [$H$] time horizon
\item [$p_i$] processing time of task $i$
\item [$p_{is}$] processing time for producing state $s$ out of task $i$
\item [$V_{iq}^{\text{min}}$] minimum batch size for processing task $i$ on unit $j$
\item [$V_{iq}^{\text{max}}$] maximum batch size for processing task $i$ on unit $j$
\item [$\rho_{is}^c$] fraction of batch of task $i$ that is composed of state $s$
\item [$\rho_{is}^p$] fraction of batch produced out of task $i$ resulting in state $s$
\item [$C_{ps}$] sale price of state $s$
\item [$C_{rs}$] raw material cost of state $s$
\item [$C_{is}$] inventory cost of state $s$
\item [$C_{stij}$] setup cost of unit $j$ for performing task $i$
\end{description}

### Variables

\begin{description}
\item [$W_{ijt}$] 1 if unit $j$ starts processing task $i$ at the beginning of time period $t$: 0 otherwise
\item [$B_{ijt}$] amount of material which starts undergoing task $i$ in unit $j$ at the beginning of time period $t$
\item [$S_{st}$] amount of material stored in state $s$, at the beginning of time period $t$
\end{description}

\begin{equation}
\left( \sum_{j \in J} \sum_{i \in I} W_{ijt} \right) - H \left( 1 - W_{ijt} \right) \leq 0 \quad \forall j \in J, i \in I, t \in T
\end{equation}
\begin{equation}
W_{ij} V_{iq}^{\text{min}} \leq B_{ijt} \leq W_{ij} V_{iq}^{\text{max}} \quad \forall j \in J, i \in I, t \in T
\end{equation}
\begin{equation}
S_{s,t} \leq C_{s} \quad \forall s \in S, t \in T
\end{equation}
\begin{equation}
S_{s,t} = S_{s,t-1} + \sum_{i \in I} \sum_{j \in J} \rho_{is}^c B_{ijt} - \sum_{i \in I} \sum_{j \in J} \rho_{is}^p B_{ijt} + R_{s,t} - D_{s,t} \quad \forall s \in S, t \in T
\end{equation}

Equation (1) is an allocation constraint which blocks time periods from the beginning of the task $i$ until it is finished ($t+p_i$) once task $i$ is allocated to be performed on unit $j$ at time period $t$. Equation (1) is trivially satisfied in case $W_{ijt} = 0$. Once task $i$ is allocated to a unit, its batch size has to be within a minimum and a maximum, as stated by Equation (2). Equation (3) is related to the available storage capacity for state $s$, whereas Equation (4) is the material balance which links the exceeding production with storage. Also, material can be consumed from storage. $R_{s,t}$ is the amount of material of state $s$ that is outsourced whereas $D_{s,t}$ corresponds to the amount of material that is allocated to fulfill demand of state $s$. The objective function has been modified to:

\begin{equation}
\max z = \sum_{s \in S} \sum_{t \in T} C_{ps} D_{s,t} - \sum_{s \in S} \sum_{t \in T} C_{rs} R_{s,t} + \sum_{s \in S} \sum_{t \in T} C_{is} S_{s,t} - \sum_{s \in S} \sum_{t \in T} C_{stij} W_{ijt}
\end{equation}

The first term of the objective function represents the income from selling products 1 and 2, the second term represents costs associated with the purchase of raw materials (feedstocks A, B and C), the third term accounts for the inventory costs of all products and finally the last term indentifies costs when a unit needs to be prepared for performing a given task.

### 5. Differential Evolution Algorithm

Differential Evolution (DE) is an optimization technique, proposed by Storn and Price [18], that belongs to the family of evolutionary computation, which differs from other evolutionary algorithms in the mutation and recombination schemes. Let the initial population chosen randomly consisting by $NP$ individuals called vectors. This population should cover the entire search space. For a problem with $N$ design variables each vector has $n$ parameters. Generally, this population is created by uniform probability distribution. In this way the population follows a natural evolution, but $NP$ does not change during the minimization process. According to Storn and Price [18], the main idea of differential evolution is to generate new individuals, called mutated vector or donor vector, by adding the weighted difference between two population random individuals to a third individual. This operation is called mutation. The new donor individual’s parameters are then mixed with the parameters of another individual randomly chosen, denoted target vector or vector to be replaced, to yield the called trial vector. This process is often referred to as crossover in the evolutionary strategy.
community. If the trial vector cost yields a lower value than the target vector cost, then the trial vector replaces the target vector in the following generation. This last operation is called selection. The process is ended when the limit of the maximum number of generations is attain or through the stagnation concept, e.g. when after several serial iterations any improvement in the population is observed.

5.1. Mutation
With the purpose to obtain the mutated vector $V^{q+1}$, let the vectors $X^q_a, X^q_b$ and $X^q_c$ mutually different and randomly chosen from the population with $NP$ individuals ($NP \geq 4$). The random indexes $a, b, c \in \{1, ..., NP\}$ are integer mutually different. In generation $q$ one pair of vectors ($X^q_a, X^q_c$) defines a difference vector ($X^q_a - X^q_c$). The perturbation rate ($F$) multiplies this difference and it is used to perturb the third vector $X^q_b$ or the best vector $X_{best}$. This factor $F$ is a real constant factor which controls the amplification of the difference vector. This process that yields the mutated vector $V^{q+1}$ can be mathematically written as:

$$V^{q+1} = X^q_a + F (X^q_a - X^q_c)$$

(6)

Figure 2a) shows the process of generating the mutated vector in the case of a two-dimensional function. All the vectors used in the operation mutation can also be visualized [19].

![Diagram](image)

Figure 2. a) The process for generating $V^{q+1}$ for two dimensional function; b) Illustration of the binomial crossover.

5.2. Crossover
Consider that for each target vector $X^q_s$, $s \in \{1, ..., NP\}$, different from indexes $a, b, c$, was generated a mutated vector $V^{q+1}$. The crossover is introduced in order to increase the diversity of the perturbed individuals. Thus, the trial vector $U^{q+1}$ is formed by:

$$u(i)^{q+1} = \begin{cases} 
  x(i)^{q+1} , & \text{if } r_i \leq CR \\
  x_s(i)^{*} , & \text{if } r_i > CR, \quad i = 1, ..., N
\end{cases}$$

(7)

where $r_i$ is $i$-th evaluation of a uniform random number generator with outcome $\epsilon [0, 1]$, $CR \in [0, 1]$ is the crossover probability and it must be supplied by user. $CR$ represents the probability of the new trial vector to inherit the variable values from mutated vector. When $CR = 1$, for example, all trial vector variables will come from mutated vector $V^{q+1}$. On the other hand, $CR = 0$, all trial vector variables will come from the target vector $X^q_s$. This crossover, developed by Storn and Price [18], is called binomial crossover operator, due to independent binomial experiments, which is executed whenever a randomly picked number $r \in [0, 1]$ is lower than the $CR$ crossover probability. Figures 1b shows the binomial crossover process with seven design variables.

After the crossover, if one or more trial vector variables are out of search space then it can be brought in the bound range as following:

$$\begin{cases} 
  \text{If } u(i) < x(i)^{\min} , \text{ then } u(i) = x(i)^{\min} , & i = 1, ..., N \\
  \text{If } u(i) > x(i)^{\max} , \text{ then } u(i) = x(i)^{\max} , & i = 1, ..., N
\end{cases}$$

(8)
where $x(i)^{\text{min}}$ and $x(i)^{\text{max}}$ are the lower and upper limits, e.g. the side constraints, respectively.

5.3. Selection
The selection is the process of producing better offspring. Unlike many other evolutionary algorithms, the DE does not use ranking and proportional selection. Instead, the cost of each trial vector $(U^{(q+1)})$ is worked out and compared with the cost of target vector $X_s^{(q)}$. If the cost of target vector is lower than that of trial vector, the target is allowed to advance for the next generation $q+1$. Otherwise, the trial vector replaces the target vector in the following generation. In other words this process can be written as:

$$
\begin{align*}
\text{If } f(U^{(q+1)}) \leq f(X_s^{(q)}) & \text{, then } X_s^{(q+1)} = U^{(q+1)} \\
\text{If } f(U^{(q+1)}) > f(X_s^{(q)}) & \text{, then } X_s^{(q+1)} = X_s^{(q)}
\end{align*}
$$

(9)

Storn and Price [18] have given some simple rules for choosing the key parameters of DE for general applications. Normally, $NP$ should be about 5 to 10 times the dimension (number of parameters in a vector) of the problem. As for $F$, it lies in the range 0.4 to 1.0. Initially $F = 0.5$ can be tried, then $F$ and/or $NP$ is increased if the population converges prematurely.

DE has been successfully applied to various fields, such as: solution of multi-objective optimal control problems with index fluctuation applied to fermentation process [20], digital filter design [21], synthesis and optimization of heat integrated distillation system [22], multi-objective optimization of mechanical structures [23], solution of inverse radiative transfer problems in two-layer participating media [24], apparent thermal diffusivity estimation of the drying of fruits [25], estimation of space-dependent single scattering albedo in radiative transfer problems [26], design of fractional order controllers [27], improved DE algorithm for the dynamic updating of the population size to reduce the number of objective function evaluations to solve optimal control problems in chemical and mechanical engineering [28], and other applications [29].

6. Results and Discussion
The aim of this paper is to determine of scheduling optimal using the DE algorithm. For this purpose the following steps are established:

- Objective function (Equation (5)), design variables (defined in the section 4).
- DE parameters: population size (500), perturbation rate - $F$ [0.1 0.5 0.8 1.0 2.0] and crossover probability - $CR$ [0.1 0.3 0.5 0.8 0.9] and DE/rand/1/bin strategy;
- Stopping criterion: maximum number of objective function evaluations equal to 10000.
- DE was run 20 times by using 20 different seeds for the random generation of the initial population.
- To compare the quality of solution, the branch and bound method, implemented in GAMS software, is used.

Figure 3 presents the boxplots considering the influence of the crossover probability and perturbation rate to solve the scheduling problem.

In these figures is possible to observe that the best solutions are found when the crossover probability and perturbation rate are equals to 0.8. These values are compatible with the reported literature, since higher values of the crossover probability and perturbation rate favor the choice of new candidates and increase the diversity of population.

Figures 4 and 5 present the Gant chart of job sequence vector on machines using DE and GAMS software.
In these figures is possible to observe that the DE algorithm is able to estimate satisfactory the scheduling optimal (1932.66) when compared with the results obtained by the branch and bound method (1932.65). It is important to emphasize that, as expected, the number of objective function evaluations necessary by the DE is much greater than those required by classical approach implemented in GAMS software.

7. Conclusion

In the present work, the Differential Evolution Algorithm was used to determine the scheduling optimal. This problem is described by 421 equations (equality and inequality), and by 361 design variables (281 continuous and 80 discrete). The computational time and number of objective function evaluations required by DE are superior to necessary by GAMS software. This result is expected due to characteristics of evolutionary technique considered. Despite the disadvantages mentioned, DE led to good results when compared with the obtained by classical approach implemented in GAMS software.

Further research work will be focused on the extension of the proposed approach to the multi-objective context.

8. Acknowledgements

The authors acknowledge the financial support provided by FAPEMIG, Fundação de Amparo à Pesquisa de Minas Gerais.

9. References


