Acta Scientiarum. Technology
ISSN: 1806-2563
eduem@uem.br
Universidade Estadual de Maringá
Brasil

Seido Nagano, Marcelo; Soriano Sampaio Januário, João Carlos
Evolutionary heuristic for makespan minimization in no-idle flow shop production systems
Acta Scientiarum. Technology, vol. 35, núm. 2, abril-junio, 2013, pp. 271-278
Universidade Estadual de Maringá
Maringá, Brasil

Acta Scientiarum

http://www.uem.br/acta
ISSN printed: 1806-2563
ISSN on-line: 1807-8664
Doi: 10.4025/actascitechnol.v35i2.12534

Evolutionary heuristic for makespan minimization in no-idle flow shop production systems

Marcelo Seido Nagano* **and João Carlos Soriano Sampaio Januário**

*Departamento de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, Av. Trabalhador Sãocarlense, 400, 13566-590, São Carlos, São Paulo, Brazil. *Author for correspondence. E-mail: drnagano@sc.usp.br*

ABSTRACT. This paper deals with no-idle flow shop scheduling problem with the objective of minimizing makespan. A new hybrid metaheuristic is proposed for the scheduling problem solution. The proposed method is compared with the best method reported in the literature. Experimental results show that the new method provides better solutions regarding the solution quality to set of problems evaluated.

Keywords: scheduling, flow shop, no-idle, makespan, evolutionary heuristic, cluster search.

Heurística evolutiva para a minimização da duração total da programação em sistemas de produção *no-idle flow shop*

RESUMO. Este artigo trata do problema de programação de operações em um ambiente de produção *no-idle flow shop*, cujo objetivo é minimizar a duração total da programação. Uma nova meta-heurística híbrida é proposta para a solução do problema. O método proposto é comparado com o melhor método reportado na literatura. Os resultados experimentais mostraram a superioridade do novo método para o conjunto de problemas tratados em relação à qualidade das soluções obtidas.

Palavras-chave: programação da produção, *flow shop*, *no-idle*, duração total da programação, heurística evolutiva, *cluster search*.

Introduction

The *flow shop* scheduling problem is a problem of work scheduling in which *n* tasks must be scheduled, in the same sequence, in a set of *m* distinct machines. A particular case of *flow shop* scheduling, called permutational, occurs when in each machine is maintained the same processing order of tasks.

The solution for the problem is to determine among the (*n*!) possible sequences of tasks, the one that optimizes some measure of scheduling performance, being that the most common consist of minimizing the total scheduling duration (*makespan*), or minimize the sum of the tasks flow times (*total flowtime*). The first relates to an efficient use of resources (machines) while the second aims to minimize the stock in processing.

This scheduling problem has been intensely studied in literature, since the first study reported by Johnson (1954) in obtaining the optimal solution for the problem with two machines.

The restrictions commonly considered in the *flow shop* scheduling problem restrain but do not exclude their practical applications in fact (DUDEK et al., 1992). One of the restrictions is that the machines that compose the *flow shop* do not suffer interruption once initiated. A representation for this problem is presented in the Figure 1.

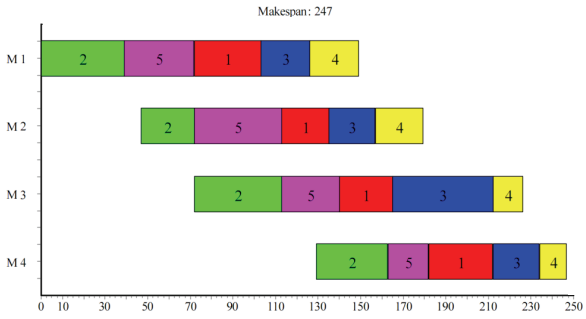

Figure 1. No-idle flow shop.

This type of problem is known as *no-idle flow shop* (*NIFS*), and occurs when the time for preparation or the cost of using the machine are relatively high, considering that to turn it on or to prepare it more times than the necessary provokes a costly process. Clear examples are the equipments used in the production of integrated circuits, by means of photolithography, a technique that creates the desired mould on the semiconductor slide.

Other examples are the ovens of ceramic cylinder that consume a large amount of natural gas when in operation, due to the thermal inertia that

makes impossible to stop or restart their functioning. In all cases, the best alternative is to determine a sequencing that considers the restriction of non-idleness in the machines. This same problem can be denoted by $F/prmu,no\text{-}idle/C_{max}$, and its computational complexity was considered as a *NP-hard* type by Tanaev et al. (1994).

Given the above, it is verified the importance and the difficulties found to solve the *NIFS* problem. This article aimed to present a new evolutionary heuristic method called HE-*NIFS*, and compare it with the most recent method proposed currently.

Heuristic methods for the *NIFS* problem

Adiri and Pohoryles (1982) were the first to address the *no-idle* problem. In this study, they also addressed the *no-wait flow shop* (*NWFS*) problem. The main contribution was the development of an algorithm which optimally solves the problem $F2/prmu,no\text{-}idle/\sum C_j$. They also showed results for the problem with more than two machines, however, in special cases of dominance among the machines.

The heuristic methods for the problem of *m*-machines were initially addressed by Woollam (1986) with the objective to minimize the *makespan*. Basically, several methods were selected from the literature, including the known method NEH. In the study, five methods were adapted, and the computational experimentation was performed with problems up to 25 tasks and 25 machines. However, for such problems, the adapted NEH method produced the best results.

Narain and Bagga (2003) studied the problem $F3/prmu,no\text{-}idle/C_{max}$. They presented an integer linear programming method and an algorithm *Branch & Bound* with limited computational results. The same problem with three machines was studied by Saadani et al. (2003). They proposed a lower limiting (*lower bound*) and an efficient heuristic method. The heuristic was compared to a previous method presented in Saadani et al. (2005) and presented better results. An important note is that this study was subsequently published in Saadani et al. (2005).

Kamburowski (2004) proposed a network representation that provided a better representation of the problem, indentifying some paradoxes resulting from the *no-idle* condition, and conducting some relationships of dominance among the machines in which the problem becomes effectively resolved.

Saadani et al. (2005) presented heuristics based on *Travelling Salesman Problem* (*TSP*) for $F/prmu,no\text{-}idle/C_{max}$. Basically, the authors modeled as distances the *makespan* of pairs of tasks. Based on the minimum distance, the heuristic applies search engines in the neighborhood, inserting the tasks, one by one, in every possible position. The heuristics possesses a complexity $O(n^3)$ and it is easily implementable. The authors tested the proposed heuristic comparing its solutions to an integer linear programming model for problems up to 17 tasks and 30 machines.

Kalczynski and Kamburowski (2005) proposed a heuristic for $F/prmu,no\text{-}idle/C_{max}$ with computational complexity of $O(n^2m)$. The heuristic was compared with Saadani et al. (2005), and presented the best results in most the evaluated problems. The authors presented also an adaptation of the NEH heuristic for the *no-idle* problem, and the results showed that the proposed method also exceeds this heuristic.

Most recently, the *NIFS* problem was once again studied by Kalczynski and Kamburowski (2007), whereby were presented special situations between the *no-wait* and *no-idle* problems. The authors presented a network representation in which the longest path lengths in the network represent the *makespan*. The networks revealed the duality between the two problems, and a graphical explanation about the condition *no-wait* and *no-idle* for *makespan* was discussed.

Baraz and Mosheiov (2008) proposed a heuristic method of two phases for $F/prmu,no-idle/C_{max}$. In the first phase, the tasks were added, one by one at the end of the current sequence, and the task that resulted in the shortest *makespan* was sequenced. This phase was performed until all the tasks had been sequenced. In the second phase, all the tasks were interchanged within the sequence obtained in the first phase. The authors concluded that the execution time of their heuristic was $O(n^2)$. However, it is worth highlighting that the authors did not consider the additional complexity of the *makespan* calculation in each stage. As this calculation has a computational complexity $O(nm)$, they showed the superiority of their proposed heuristic in comparison to the results obtained by Saadani et al. (2005), however they did not present a comparison to the heuristic of Kalczynski and Kamburowski (2005).

Pan and Wang (2008a and b) in two similar studies, proposed a procedure that accelerates the search in the neighborhood of insertion and reduces the computational complexity from $O(n^3m)$ to $O(n^2m)$. This acceleration is based on procedures presented in Taillard (1990) for the same search in the neighborhood, but for the traditional *flow shop* problem. Both algorithms used a method of search called *Iterated Greedy* (RUIZ; STÜTZLE, 2007).

The authors used the problems of Taillard (1993) of the classical *flow shop* for the *no-idle* problem. In the two studies were compared the proposed methods to the heuristics of Baraz and Mosheiov (2008) and Kalczynski and Kamburowski (2005). The results indicated the superiority of the proposed methods; however the results were not compared among them.

Finally, an extensive computational experimentation was carried out by Ruiz et al. (2009) for the *no-idle flow shop* problem. The authors performed two computational experiments: the first was the evaluation of deterministic heuristic methods; and the second was the evaluation of metaheuristic methods. As for the first experiment, were evaluated nine heuristic methods adapted to the *NIFS* problem as listed below:

- NEH of Nawaz et al. (1983) with acceleration of Pan and Wang (2008a and b), with computational complexity of $O(n^2m)$;
- The original NEH without acceleration, referred to as NEH_{na}, with computational complexity of $O(n^3m)$;
- SGM of Saadani et al. (2005), as computational complexity of $O(n^3)$;
- KK of Kalczynski and Kamburowski (2005), with computational complexity of $O(n^3m)$;
- GH_BM of Baraz and Mosheiov (2008), with computational complexity of $O(n^3m)$;
- The new GH_BM2 proposed method with acceleration, with computational complexity of $O(n^2m)$;
- GH_BM2 without acceleration referred to as GH_BM2_{na}, with computational complexity of $O(n^3m)$;
- FRB3 of Rad et al. (2009), with computational complexity of $O(n^3m)$;
- $FRB4_k$ of Rad et al. (2009) with k values of 4 and 12 ($FRB4_4$ and $FRB4_{12}$), with computational complexity of $O(kn^2m)$ or $O(n^2m)$.

In the second computational experimentation were evaluated four metaheuristic methods adapted to the *NIFS* problem:

- HDPSO of Pan and Wang (2008a);
- DDELS of Pan and Wang (2008b);
- FRB5 of Rad et al. (2009);
- IG_{LS} of Ruiz and Stützle (2007).

The results obtained by the authors in the first computational experimentation indicated that the methods FRB3 and GH_BM2 had the best performances. The first regarding the relative average deviation, and the second regarding the results quality and the computing time.

The results obtained in the second experimentation pointed out that the IG_{LS} had results significantly better than the HDPSO.

Finalizing according to the literature review performed and reported in this study, it can be concluded that the adapted IG method, as well as the improved proposed method GH_BM2, together with the heuristics of Rad et al. (2009) are the best existing methods for the *NIFS* problem with the *makespan* minimization criterion.

Material and methods

New evolutionary heuristic

The evolutionary heuristic proposed in this research is a hybridization of the *Cluster Search* (*CS*) presented by Ribeiro Filho et al. (2007) with the IG_{LS} method proposed by Ruiz and Stützle (2007).

In this evolutionary process the stopping criterion is determined by the computing time (*elapsed time*) of the problem (MORAES; NAGANO, 2012). The time limit will be adopted as being $n(m/2)T$ milliseconds, where T is a regulatory parameter of the stopping criterion. With a stopping criterion proposed in that way the computational time will be proportional to the number of tasks and machines. Thus, the greater the (n; m) problem, the greater will be the time available for the iterations.

The quality of the solution obtained by the *CS* is mostly dependent on the initialization processes present in its structure. Among these initializations exists the initial population initialization and the *clusters* initialization.

The initial population is formed by a vector of k positions, where each position is a solution. The size of this vector is defined as being the number of individuals possible to be generated in a time limit, being 500 the maximum value for k. Defining this time limit to generate the population as being 10% of the computing time (*elapsed time)* for the problem already defined, the initial population will be in the maximum of 500 individuals or a limited value by the 10% of the *elapsed time*. In this way it is understood that in the maximum of 10% of the computing time for the problem will be spent to initialize the population.

In order to ensure the quality of the individuals generated in the population, it is proposed a combination of the heuristic NEH of Nawaz et al. (1983) and the method *Iterated Greedy* (*IG*) (RUIZ; STÜTZLE, 2007), in which are generated solutions (sequences) through two phases: destruction and construction. In the phase of destruction some elements of the sequence are removed to be inserted subsequently, reconstructing the sequence. The motivation for using a combination of methods in the

evolutionary heuristic (HE-*NIFS*) proposed, was the search for a greater variety of solutions with better quality in the process of initialization.

In the population initialization, the first individual inserted in the population is generated by the procedure NEH. All the other individuals are generated by the method *Iterated Greedy* (*IG*), applying the phases of destruction and construction to an individual already created. Thus, the first individual is generated by the procedure NEH, the second will be generated by the destruction and subsequent construction of the first individual. The third will be found by the process of destruction and construction of the second, and so on. Following this process, the forming of an individual requires other already formed, that is, the process is recursive, except the first individual generated by the heuristic NEH.

This procedure aims to provide a greater diversity of solutions in the process of initialization, without having to apply a simply random process, which would lead to the loss of quality of the generated solutions in the initial population.

The evaluation of the generated individuals is made directly by the optimization criterion for the *makespan*, and the evaluation is responsible for maintaining the population ordered, with the best individual (that individual with the shortest *makespan*) occupying the first position. This same procedure does not allow the insertion of repeated individuals in the population.

A procedure of initialization of *clusters* is created to seize the good individuals from the initial population. In the same way as the population, the *clusters* created will be submitted to the process of evaluation to order them. Once completed its initialization, the *clusters* evaluated as having their *makespan* among the best 1/3, within the all generated, will pass a process of local search (LS1) in view of a better selection of the space of solutions to be worked by the method *Cluster Search*.

The *clusters* initialization scans the population from the best to the worst individual, generating new *clusters* or assimilating the individuals into the *clusters* already created. New clusters are created when the individual in question is not found within the radius ($r = 0.85n$) of any *clusters* already existing, where n is the number of tasks of the sequence. The assimilation of an individual is made in the *cluster* whose center this individual is found closer.

The measurement of distance between the permutations of the individuals p_i and the centers of the clusters c_j was adopted as being the amount of necessary exchanges to transform p_i into c_j. The process of generation of these initial *clusters* ends when the entire population is scanned or when generated the maximum number of *clusters*, in the present study it was determined as being 200.

A *Cluster Search* characteristic is the evolutionary behavior, in which after a start with several *clusters* created, its number is slowly reduced, prevailing the *clusters*, in the regions of the space, with the best solutions.

The assimilation process of an individual in a *cluster* has as basis the *Path Relinking* process (GLOVER, 1996). From an individual p_i, successive exchanges of a pair of tasks will be made until its sequence becomes identical to the sequence of the *cluster* center. At every exchange a new sequence is generated and evaluated. The pair of tasks chosen to be exchanged is that pair that produces the sequence with the best evaluation at every step. In the end, if the own assimilated individual p_i, or the best sequence found in the successive exchanges, has the best evaluation than the *cluster* center, this becomes the new center of this *cluster*.

At every attempt to generate a new individual two others are selected in the population, one of them among the best 10%, called base individual, and other among all the individuals in the population, called guide. A crossover process, or recombination, known as BOX (*Block Order Crossover*) proposed by Syswerda (NAGANO et al., 2012), is used to generate the new individuals. In this technique, the parent individuals are combined through random copy of blocks of genes of both parents, which results in the generation of a single child containing part of the parent's heritage. In this study, the child was generated with 75% of genes from the parent 1 (base) and 25% from the parent 2 (guide).

After recombination, the new individual has 60% of probability to go through a process of improvement in the form of a local search. This improvement can be done by two types of local search, LS1, with probability of 40% to occur, or LS2, with 20%.

Both the LS1 and the LS2 are hybrid processes that use two types of neighborhood, the permutation and the insertion. Both are detailed in Figures 2 and 3, being the Figure 2 about the LS1 and the Figure 3 about the LS2. Another relevant characteristic in these searches is the random character, because both are made through the random choice and without repeating the tasks to be worked in the neighborhood. Such a procedure was proposed, once again, to guarantee a greater diversity of solutions.

Local Search LS1(π)

```
end := 0;
while (end = 0) and (time < 10% of the total
processing time);
    sp ← best sequence π' or the
         sequence itself π after the
         application of the
         neighborhood of
         permutation in the sequence
         π;
    si ← best sequence π' or the
         sequence itself π after the
         application of the
         neighborhood of insertion
         in the sequence π;
    if (Makespan (sp) < Makespan (si))
    and (Makespan (sp) < Makespan
    (π));
         π ← sp;
    otherwise (Makespan (si) <
    Makespan (sp)) and (Makespan (si)
    < Makespan (π));
         π ← si;
    otherwise end := 1;
```

Figure 2. Local Search LS1.

In the neighborhood of permutation all the possible pairs of tasks of the sequence are exchanged, thereby generating *n(n-1)/2* new sequences. In the neighborhood of insertion each task is removed from its position and inserted in all other possible positions, being laterally moved to fill the position left by it, thereby generating $(n-1)^2$ new sequences.

Every new individual, most times improved by one of the two local searches, and which is not identical to any individual already belonging to the population, is inserted into the population in a position relating to its evaluation, thereby causing the removal of the worst individual present in the population up to the moment. Thus, the evolutionary process eventually updates the population at every new generated individual.

The new individuals inserted successfully in the population are assimilated by the *cluster* whose center they are closer, or generate new *clusters*.

Local Search LS2(π)

```
end := 0;
while (end = 0) and (time < 10% of the total
processing time);
    si ← best sequence π' or the
         sequence itself π after the
         application of the
         neighborhood of insertion
         in the sequence π;
    if Makespan (si)< Makespan (π);
         π ← si;
         sp ← best sequence π' or
              the sequence
              itself π after the
              application of the
              neighborhood of
              permutation in
              the sequence π;
         if Makespan (sp)<
         Makespan (π);
              π ← sp;
    otherwise sp ← best sequence π' or
              the sequence itself
              π after the
              application of the
              neighborhood of
              permutation in π;
         if Makespan (sp)<
         Makespan (π);
              π ← sp;
    otherwise end := 1;
```

Figure 3. Local Search LS2.

Aiming to select even more the space of solutions to be worked by the *Cluster Search* a new local search (LS2) will be performed in the *clusters* classified as belonging to the best 1/3 portion, by the

evaluation criterion for the *makespan*. This second local search will be performed when the computing time of the method HE-*NIFS* exceeds 50% of the computing time proposed by the stopping criterion (*elapsed time)*. This local search will be performed only once, not being repeated until the proposed method ends.

During all the process, the best *cluster*, i.e., that whose center has the best evaluation will be maintained safe. Being this the solution presented by the *Cluster Search*.

For a better understanding of the *Cluster Search*, the Figure 4 presents the procedure in detail.

***Cluster Search*(π)**

tls2 := 0;
while time < 10% of the total processing time;
  for $i \leftarrow$ 1 up to n;
    P(i) $\leftarrow$ individual generated in the population initialization;
  Initial population ordering;
  for $i \leftarrow$ 1 up to n;
    P(i) can generates a new cluster or suffer assimilation (Clusters initialization);
while time < Total Time of Processing;
  creation of a new individual (Box Recombination);
  improvement of the new individual in 60% of cases;
  assimilation or creation of a new cluster through the new individual;
  if time > 50% of the total processing Time
    tls2 := 1;
    the 1/3 best clusters will pass through the local search LS2;

Figure 4. *Cluster Search* procedure.

Results and discussion

Aiming to evaluate the performance of the method HE-*NIFS*, the method was compared with the best method reported in the literature, known as IG_{LS} proposed by Ruiz et al. (2009). All methods were coded into language C and processed in a microcomputer Intel Xeon 3GHz and 32GB of RAM.

It was used the Taillard database made up by 10 problems for each class, totaling 120 test problems.

The statistics used to evaluate the methods performance were the Percentage of Success (PS) and the Relative Deviation (RD).

The first statistics is defined by the quotient between the total number of problems for which the method obtained the best solution, and the total number of problems resolved. The second quantifies the relative deviation (RD_h) that the method h obtains in relation to the best solution for a same problem, being calculated by:

$$RD_h = \frac{(M_k - M.)}{M.}$$

where:

M_h: *makespan* obtained by the method h;

$M.$: best *makespan* obtained by the methods for a determined problem.

Results analysis

The methods IG_{LS} and HE-*NIFS* were evaluated considering the stopping criterion $n(m/2)T$ for $T \in \{50, 250, 500, 750, 1000\}$.

The Table 1 presents the summarized results for the values of percentage of success and percentage of relative average deviation for the methods IG_{LS} and HE-*NIFS*. The same results are presented individually in the Figures 5, 6 and 7.

Table 1. Percentage of success and percentage of relative average deviation for the methods IG_{LS} and HE-NIFS.

T	IG_{LS}	HE-NIFS
50	64.17⋆	64.17
	1.1844⋆⋆	1.1829
250	69.17	66.67
	0.9338	0.8407
500	63.33	79.17
	0.8706	0.6802
750	77.50	62.50
	0.8899	0.7083
1000	70.00	72.50
	0.7843	0.3427

⋆Percentage of success; ** Percentage of relative average deviation.

The Figure 5 shows the percentage of success for the methods IG_{LS} and HE-*NIFS* for the total of problems evaluated according to the variation of the parameter T.

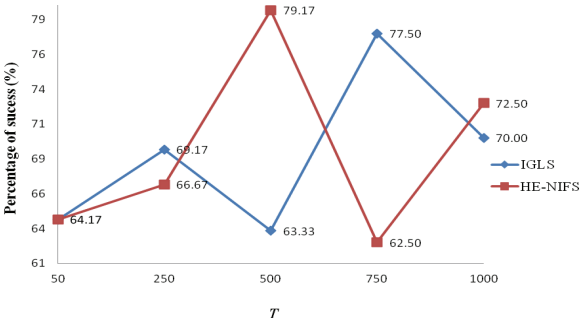

Figure 5. Percentage of success for the methods IG_{LS} and HE-*NIFS*.

It was observed that none of the evaluated methods presented predominant trend. Both methods oscillated according to the variation of the parameter *T*.

The Figure 6 presents the percentage of relative average deviation for the methods IG_{LS} and HE-*NIFS* for the total of problems evaluated according to the variation of the parameter *T*.

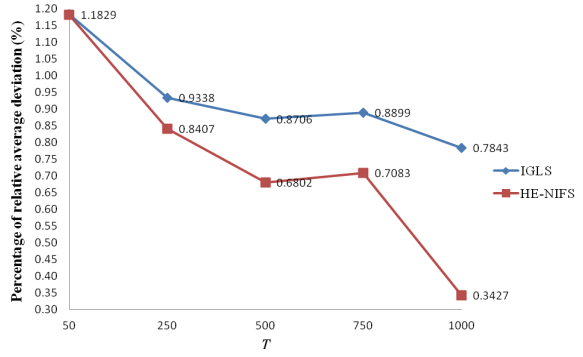

Figure 6. Percentage of relative average deviation for the methods IG_{LS} and HE-NIFS.

In the Figure 6 it was observed a gradual trend for reduction of the relative average deviation for the method HE-NIFS as the value of the parameter *T* increased. For this reason, according to the Figure 6, it can be concluded that the method HE-*NIFS* obtained solutions with better quality in comparison to the method IG_{LS}.

The Figure 7 presents the statistical significance of the difference between the heuristics IG_{LS} and HE-*NIFS* for the values of *T* referent to the relative deviations. In the figure it was calculated the mean value of heuristic pairs corresponding to 95% confidence intervals.

The Figure 7 pointed out that the quality of the solution between the methods IG_{LS} and HE-*NIFS* presented significant differences as the values of *T* were evaluated. The method HE-*NIFS* has proved to be more stable in relation to the quality of the obtained solutions, on the other hand, the IG_{LS} method presented a growing increase in the confidence interval variation, reflecting in solutions of lower quality according to increasing values of *T*.

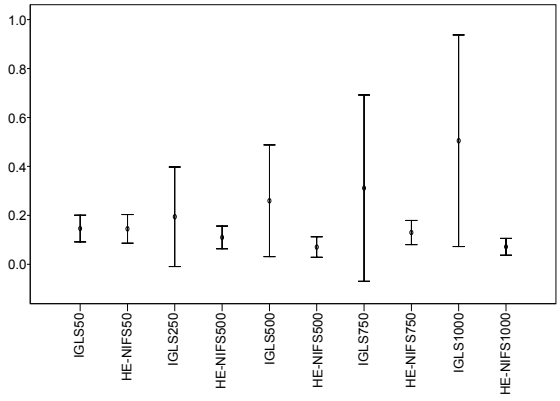

Figure 7. Mean and confidence intervals (95%) for the pairs of algorithms evaluated.

The results for the computational experimentation in the present study are motivating, despite the percentage of success obtained by the method HE-*NIFS* not being far superior when compared to the method IG_{LS}, the results indicate that the method HE-*NIFS* obtains solutions of better quality presenting little variation when compared to IG_{LS}. Thus, it was verified the superiority of the new method HE-*NIFS* in relation to the achievement of solutions of better quality comparing the set of evaluated problems.

Conclusion

The experimental results showed that the heuristic method HE-*NIFS* presented equal performance considering the percentage of success, and superior regarding the relative deviation (quality of the obtained solutions) in comparison to the method IG_{LS}, which is considered the best method currently available. Thus, it can be affirmed that the method HE-*NIFS* is an alternative method with highest quality for the *no-idle flow shop* scheduling problem with criterion of minimizing *makespan*.

The fact that HE-*NIFS* presented solution of better quality is guaranteed by the great diversity of initial solutions allowed by the *Cluster Search* that provides a better selection of space of final solutions.

Although the initial results presented had already been considered satisfactory, the method HE-NIFS can still be improved by finding optimal parameters, through an exhausting computational experimentation, in which can be evaluated the parameters variations and their respective provided solutions. This process will constitute a continuity

of the present study with the goal of applying the technique of design of experiments (Design of Experiments - DOE).

Acknowledgements

The authors thank to Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Brazil for financial support.

References

ADIRI, I.; POHORYLES, D. Flowshop no-idle or no-wait scheduling to minimize the sum of completion times. **Naval Research Logistics**, v. 29, n. 3, p. 495-504, 1982.

BARAZ, D.; MOSHEIOV, G. A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time. **European Journal of Operational Research**, v. 184, n. 2, p. 810-813, 2008.

DUDEK, R. A.; PANWALKAR, S. S.; SMITH, M. L. The lessons of flowshop scheduling research. **Operations Research**, v. 40, n. 1, p. 7-13, 1992.

GLOVER, F. Tabu search and adaptive memory programing: Advances, applications and challenges. In: **Interfaces in computer science and operations research**. Dordrecht: Kluwer Academic Publishers, 1996.

JOHNSON, S. M. Optimal two- and three-stage production schedules with setup times included. **Naval Research Logistics Quarterly**, v. 1, n. 1, p. 61-68, 1954.

KALCZYNSKI, P. J.; KAMBUROWSKI, J. A heuristic for minimizing the makespan in no-idle permutation flow shops. **Computers and Industrial Engineering**, v. 49, n. 1, p. 146-154, 2005.

KALCZYNSKI, P. J.; KAMBUROWSKI, J. On no-wait and no-idle flow shops with makespan criterion. **European Journal of Operational Research**, v. 178, n. 3, p. 677-685, 2007.

KAMBUROWSKI, J. More on three-machine no-idle flow shops. **Computers and Industrial Engineering**, v. 46, n. 3, p. 461-466, 2004.

NAGANO, M. S.; SILVA, A. A.; LORENA, L. A. N. A new evolutionary clustering search for a no-wait flow shop problem with set-up times. **Engineering Applications of Artificial Intelligence**, v. 25, n. 6, p. 1114-1120, 2012.

NARAIN, L.; BAGGA, P. C. Minimizing total elapsed time subject to zero total idle time of machines in n × 3 flowshop problem. **Indian Journal of Pure and Applied Mathematics**, v. 34, n. 2, p. 219-228, 2003.

NAWAZ, M.; ENSCORE JR., E. E.; HAM, I. A heuristic algorithm for the m-machine, n-Job flow-shop sequencing problem. **OMEGA, The International Journal of Management Science**, v. 11, n. 1, p. 91-95, 1983.

MORAES, M. B. C.; NAGANO, M. S. Cash balance management: A comparison between genetic algorithms and particle swarm optimization. **Acta Scientiarum. Technology**, v. 34, n. 4, p. 373-379, 2012.

PAN, Q.-K.; WANG, L. No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. **International Journal of Advanced Manufacturing Technology**, v. 39, n. 7-8, p. 796-807, 2008a.

PAN, Q.-K.; WANG, L. A novel differential evolution algorithm for noidle permutation flow-shop scheduling problems. **European Journal of Industrial Engineering**, v. 2, n. 3, p. 279-297, 2008b.

RAD, S. F.; RUIZ, R.; BOROOJERDIAN, N. New high performing heuristics for minimizing makespan in permutation flowshops. **OMEGA, the International Journal of Management Science**, v. 37, n. 2, p. 331-345, 2009.

RIBEIRO FILHO, G.; NAGANO, M. S.; LORENA, L. A. N. Evolutionary clustering search for flowtime minimization in permutation flow shop. **Lecture Notes in Computer Science**, v. 4771, p. 69-81, 2007.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 177, n. 3, p. 2033-2049, 2007.

RUIZ, R.; VALLADA, E.; FERNÁNDEZ-MARTÍNEZ, C. Scheduling in flowshops with no-idle machines. In: CHAKRABORTY, U. K. (Ed.). **Computational intelligence in flow shop and job shop scheduling**. Berlin: Springer, 2009. p. 21-51.

SAADANI, N. E. H.; GUINET, A.; MOALLA, M. Three stage no-idle flow-shops. **Computers and Industrial Engineering**, v. 44, n. 3, p. 425-434, 2003.

SAADANI, N. E. H.; GUINET, A.; MOALLA, M. A travelling salesman approach to solve the F/no−idle/Cmax problem. **European Journal of Operational Research**, v. 161, n. 1, p. 11-20, 2005.

TAILLARD, E. Some efficient heuristic methods for the flow shop sequencing problem. **European Journal of Operational Research**, v. 47, n. 1, p. 65-74, 1990.

TAILLARD, E. Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, n. 2, p. 278-285, 1993.

TANAEV, V. S.; SOTSKOV, Y. N.; STRUSEVICH, V. A. **Scheduling theory**. Multi-stage systems. Dordrecht: Kluwer Academic Publishers, 1994.

WOOLLAM, C. R. Flowshop with no idle machine time allowed. **Computers and Industrial Engineering**, v. 10, n. 1, p. 69-76, 1986.

Received on February 18, 2011.
Accepted on May 10, 2012.