Revista de Matemática: Teoría y Aplicaciones 2002 **9**(1) : 35–46
cimpa – ucr – ccss issn: 1409-2433

A TABU SEARCH APPROACH FOR THE WEIGHTED TARDINESS WITH SEQUENCE-DEPENDENT SETUPS IN ONE-MACHINE PROBLEM

Ricardo P. Beausoleil[*]

Recibido/Received: 15 Febrero/February 2001

Abstract

In this paper, a Tabu Search Approach for the weighted tardiness single machine problem with sequence-dependent setups is proposed. The main contribution is the balance obtained between intensification and diversification strategies. The strategy of combine large step optimization, frequency-based memory, intensification by decomposition supplementing this with an additional intensification using path relinking produce good solutions with a low computational cost. Our Tabu Search approach is compared with a re-start method that employs the all-pairs neighborhood. Results of computational experiments are reported for a set of randomly generated test problems.

Keywords: Tabu Search, scheduling problems, weighted tardiness, sequence depend-setups.

Resumen

En este artículo, se propone un enfoque basado en Búsqueda Tabú para el problema de una sola máquina, con retardo ponderado, con puestas a punto que dependen de la sucesión. La principal contribución es el balance obtenido entre las estrategias de intensificación y diversificación. La estrategia de combinar amplios pasos de optimización, memoria basada en la frecuencia, intensificación por descomposición con una intensificación adicional que usa religamen de caminos, produce buenas soluciones con un costo computacional bajo. Nuestro enfoque de Búsqueda Tabú es comparado con el método de inicio múltiple que emplea el vecindario de todos los pares. Se reportan resultados de experimentos computacionales para un conjunto de problemas test generados aleatoriamente.

[*] Centro de Matemática y Física Teórica, Calle E # 309 esq. a 15, Vedado, Ciudad de La Habana, C.P. 10400, Cuba. Fax: +(537) 33 33 73. E-Mail: rbeausol@cidet.icmf.inf.cu

Palabras clave: Búsqueda Tabú, problemas de calendarización, retardo ponderado, puestas a punto que dependen de la sucesión.

Mathematics Subject Classification: 90C27

1 Introduction

Tabu Search is a meta-heuristic that makes use of historical information. The historical information is kept in three kinds of memory functions: the short-term,the intermediate and long term memory function. Short term memory function has the task of memorizing certain complete solutions or attributes of the search process of the recent past and it is incorporated in the search via one or more tabu lists (recency-based memory). Frequency-based memory is fundamental to longer term considerations, and provides a type of information that complements the information provided by recency-based memory. Two highly important components of tabu search are intensification and diversification strategies. Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good.

Intensification strategies generate "neighbors" by either grafting together components of good solutions or by using modified evaluation strategies that favor the introduction of such components into the current (evolving) solution. The diversification strategies lead the search process to examine unvisited regions. The objective of a diversification is to produce a new starting point, from which the search may continue.

In the approach to obtain a new starting point, two kind of diversifying strategies can be distinguished, strategies using a restart method and strategies using a method which lead the search to a new regions in an iterative fashion. The letter kinds of diversification methods often (though not always) provide advantages over restart methods, [8].

In our study we have implemented a diversification as an iterative method, using frequency counts, and a large-step procedure. Large step optimization consists of three main routines, a large-step phase, a small-step phase and an accept/reject test, where all three are consecutively executed for certain number of large step iterations. This procedure should make a large modification in the current solution to drive the search to a new region and, at the same time, to perform some kind of optimization to obtain a solution not too far away from a local (or global) optimal schedule.

Additional search intensification can be achieved using path relinking, a strategy proposed in connection with tabu search, which has been rarely used in actual implementation. One of the few path relinking implementations appears in Laguna and Martí [1]. Path Relinking offers a useful integration of intensification and diversification strategies. This approach generates new solutions by exploring trajectories that connect elite solutions - by starting from one of these solutions, called an initiating solution, and generating a path in neighborhood space that leads toward the other solutions, called guiding solutions.

2 A case of study

The problem to be discussed in this paper is the scheduling of n jobs on a single machine where the goal is to minimize the total weighted tardiness, with sequence-depend setup times. The machine is able to handle exactly one job or operation at any qiven time, and each such job or operation must be processed to completion once it is started (i.e., the model is non-preemptive). The only decisions to be made are sequencing and release timing of jobs or operations from the input queue to the resource. (Only one such queue exist.)

The machine processes jobs one at a time serially, the resource is available over the scheduling interval t_s to t_e, n-single operations jobs arrive over the interval, the job i has a processing time p_i, a ready time r_i, a due date d_i, sequence-depend setups are represented by a matrix $T_{i,j}$ the objective function is to minimize total weighted tardiness.

The "lateness" of the job i is the amount of time (positive or negative) by which the completion of activity i exceeds its due date, $L_i = C_i - d_i$. Tardiness is defined as rectified lateness, i.e; the lateness of job i if it is positive. If the lateness of job i is not positive, the tardiness is zero: $T_i = max\{0, L_i\}$. Tardiness, reflects the fact that, in many situations, distinct penalties and other costs will be associated only with positive lateness. Minimum weighted tardiness $\sum_i w_{T_i} Ti$ is a useful objective, but problems using this objective are very difficult to solve exactly. The static form of this problems was first presented by McNaughton [2]. In this problem we need only consider permutations of the n jobs to find an optimal schedule. However, even for modest-size problems, complete enumeration is not computationally feasible since it requires the evaluation of $n!$ sequences.

A number of approaches have been suggested for solving this problem without performing an explicit enumeration.Held and Karp [3] and Lawler [4] present dynamic programming formulations which require the consideration of 2^n possible subsets of the n jobs. Although much more efficient than complete enumeration, this method is still computationally infeasible for modest-sized problems (e.g. a 20-job problem requires the storage of more than a million numbers). More recently several techniques were developed, OPT-like rules came in the late 1970s and early 1980s,while bottleneck dynamics started in the early 1980s. A number of newer mathematical methods have also appeared: beam search, simulated annealing, genetic algorithms, tabu search and others. This paper presents one heuristic, based in TS methodology to solve the weighted tardiness problem for one-machine scheduling with sequence-depend setups.

3 Adaptive Approach Heuristic

3.1 Neighborhoods

Two methods of generating neighborhoods are presented in this paper, an adjacent pairwise interchange operation where two adjacent jobs are interchange and a general pairwise interchange operation that considers the neighborhood generated by every possible pairwise interchange. A general three-move to the left or right is used in order to incorporate a perturbation move.

3.2 Memory Structures

The main components of tabu search algorithms are memory structures, in order to trace and control the evolution of the search. TS maintains a selective history of the states encountered during the search, and replaces $N(s)$ by a modified neighborhood $N^*(s)$. In our implementation we use recency and frequency based memories. Associated with these memories is a fundamental memory structure called the tabu list, in our case based in an attribute memory, that is, we take account,of the identity of the pair of elements that change positions. In order to maintain the tabu list, we use an array *tabuend*(e), where e ranges over the attributes (see Glover and Laguna, [7]), which in our case consist of the positions of jobs.

Our implementation uses a complementary tabu memory structure. Associated with the array *tabuend* we have the array $frequencycount$, which identifies the distribution of the move attributes, i.e, the numbers of times that positions of jobs have been exchanged. We use these counts to diversify the search. Our algorithm uses the frequency information to penalize non-improving moves by assigning a penalty to swaps of position pairs with greater frequency.

Our implementation uses a complementary tabu memory structure. Associated with the array *tabuend* we have the array $frequencycount$, which identifies the distribution of the move attributes, i.e, the numbers of times that positions of jobs have been exchanged. We use these counts to diversify the search. Our algorithm uses the frequency information to penalize non-improving move by assigning a penalty to swaps of position pairs with greater frequency.

This implementation uses a tabu restriction that forbids swaps based in one array that records a tabu tenure for each position of jobs separately, making the move tabu inactive, if one of the selected attribute is tabu inactive. This implementation increases the percentage of available move that receive the tabu classification, permitting a small tabu tenure, whose size is $tabutenure = 3$.

3.3 aspiration criteria

Aspiration criteria are introduced in tabu search to determine when the tabu restriction can be overridden, thus removing a tabu classification. In our tabu search implementation we use a global form of aspiration by objective, that is, a move aspiration is satisfied, permitting s^{trial} to be a candidate for selection, if $c(s^{trial}) < bestcost$.

3.4 Intensification and Diversification

The diversification stage encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. Intensification strategies are based on modifying rules to encourage move combinations and solution features historically found good.

In order to implement these strategies in our TS procedures, we use a frequency measure that consists of the numbers of times that the jobs occupies every one of the positions in good schedules, this is a so called residence frequency. In a case of an intensification strategy, a high residence frequency may indicate that one attribute is highly attractive if the frequency refers to high quality solutions, and in the diversification we can use this to modifies evaluation to drive the process to new regions of the solutions space.

Our approach uses two phases, one of diversification and another of intensification diversification. In the diversification phase we use a large step optimization with short-term tabu search strategy.

The intensification strategy is an intensification by decomposition, where restrictions are imposed on part of the solution structure in order to allow a more concentrated focus on the other parts of the structure. In our case, the intersection of the jobs with greatest frequency of occupation in every position and the current elite solution is fixed in a trial solution and the others jobs with greatest frequency are highly incentives and a general pairwise interchange is applied. We use a function that creates a penalty if a trial solution not improve the current solution and creates an incentive in other case.

Denote $value = c(s^{now}) - c(s^{trial})$ then, our penalize/incentive function is

$$PI = value * (1 + (F(i,p) + F(j,q))/MaxEvent))$$

where $F(i,p)$ and $F(j,q)$ denote the frequency measure of the job i, j over the positions p and q in a subsequence of elite solutions and $MaxEvent$ is the total number of improving move.

3.5 Large-step optimization

In the procedure TS-LS we combine large-step optimization with a tabu search approach, following a design applied to a job shop scheduling problem by H. Ramalhinho [10].

The large-step optimization consists of three main routines: a large-step phase, a small-step phase, and an accept/reject test, which all three are consecutively executed for a certain number of iterations.

An important part of the large-step optimization methods is the large-step phase, that make a large modification in the current solution to drive the search to a new region.

Here we use a general 3-way interchange to left or right as the large step phase and, as small step a simple tabu search approach.

4 Intensification with Path Relinking

Path Relinking has been suggested as an approach to integrate intensification and diversification strategies [8]. This approach generates new solutions by exploring trajectories that "connect" high-quality solutions - by starting from one of the solutions, called an initiating solutions, and generating a path in neighborhood space that leads toward the others solutions, called guiding solutions. In our implementation, the path relinking strategy is used with the goal of strengthening the intensification phase. During path relinking, the main goal is to incorporate attributes of the guiding solution while at the same time recording the objective function values. In the current development, the procedure stores a small set of the elite solutions to be used as reference points (the best three solutions found during the search). The relinking process implemented in our search may be summarized as follows:

- The set of elite solutions is constructed during the intensification and diversification phases (the best three solutions).
- Designate a subset of the best solutions to be reference solutions (three different subsets are generated, all combinations of two solutions) taking as guide solutions the best in terms of the objective function.

  Incorporate attributes of guiding solutions by inserting and exchanging moves. In our implementation s is a permutation of numbers of jobs, that is, $s = (s_1, s_2, \ldots, s_n)$ then, we define $Insert(s_q, p)$ to consist of deleting s_q from its current position q to be inserted in position p.

$$\begin{aligned} s' &= (s_1, \ldots, s_{p-1}, s_q, s_p, \ldots, s_{q-1}, s_{q+1}, \ldots, s_n) \text{ for } p < q, \\ &= (s_1, \ldots, s_{q-1}, s_{q+1}, \ldots, s_p, s_q, s_{p+1}, \ldots, s_n) \text{ for } p > q. \end{aligned}$$

  and $Exchange(s_p, s_q)$ as was described in pairwise interchange, that is:

$$s' = (s_1, \ldots, s_{p-1}, s_q, s_{p+1}, \ldots, s_{q-1}, , s_p, s_{q+1}, \ldots, s_n)$$

- Neighborhood for Path Relinking: two neighborhood were considered to our path relinking, if p is the position that s_q occupies in the guide solution, then

$$\begin{aligned} N_1 &= \{s' : Insert(s_q, p), p, q \in \{1, 2, \ldots, n\}\} \\ N_2 &= \{s' : Exchange(s_p, s_q), p, q \in \{1, 2, \ldots, n\}\} \end{aligned}$$

- In conjunction with these neighborhoods, a strategy is introduced to obtain newly points, an oscillation strategy between this neighborhood is used.

4.1 TS pseudo code

Here formally we describe our tabu search approach.

General pseudo-code for the proposed heuristic.

1. Apply Large Step with short-term TS (diversification Approach).
2. Apply Intensification/Diversification TS Strategy.
3. Apply Path Relinking Strategy.

Large step with Tabu Search approach.

```
procedure LS-TS;
$iter = 0$;$numimproving = 0$;
generate starting solution in EDD order;
initialize tabulist and frequency move matrix;
```

```
$bestcost = best\, c(s^{now})$;
for 1 to 300 do;
  repeat;
    for 1 to n-1 do;
        generate a move via adjacent pairwise interchange;
        select$(p, q | tabuend[p] < iter\, or\, tabuend[q] < iter$
        or $c(s^{trial}) < bestcost)$;
        if non-improving trial move penalize $c(s^{trial})$;
        choice the best $s^{trial}$;
    endfor;
    $s^{now} = best\, s^{trial}$;
    $c(s^{now}) = c(best\, s^{trial})$;
    if improving move then
      $bestcost = c(s^{now})$;
      $s^{best} = s^{now}$;
      update frequency matrix;
      update frequency move and tabu tenure;
      $nonimprove = 0$;
      $lastimproving = current\ iteration$;
      $numimproving = numimproving + 1$;
    else $nonimprove = nonimprove + 1$;
  until $nonimprove = 20 * numimproving/lastimproving$;
  generate a perturbation move via general 3-way
endfor;
```

Tabu Search approach with Intensification/Diversification Strategy.

```
procedure TS-ID;{
$iter = 0$;
tabulist empty;
repeat;
  $iter = iter + 1$;
  $bestcost = best\, c(s^{now})$;
  repeat;
    $counter = 0$;
    repeat;
      $counter = counter + 1$;
      generate a move via general pairwise interchange;
      $select(p, q|$ $(tabuend[p] < iter\, or\, tabuend[q] < iter)$
                  and $tabupenalty[p] < iter\, and\, tabupenalty[q] < iter$;
                  or $c(s^{trial}) < bestcost)$
                  incentive or penalize $c(s^{trial})$;
      choice the best $s^{trial}$;
    until $counter = n(n-1)$;
  $s^{now} = best\, s^{trial}$;
  $c(s^{now}) = c(best\, s^{trial})$;
  if improving move
    then
        update $bestcost\, and\, s^{best}$;
        update list of elite solution;
        update frequency matrix;
  $movep = p; moveq = q$;
  $tabutenure[movep] = tabutenure[movep] + iter$;
  $tabutenure[moveq] = tabutenure[moveq] + iter$;
until $iter = 2000$;
```

5 Re-start method (RS) as Benchmark

Tan and Narasimhan [15] chose the RS technique as a baseline benchmark for conducting comparisons with simulated annealing (SA) approach they proposed. Also Franca, Mendes and Moscato [14] proposed this method as benchmark in your work with memetic algorithm.

We use as re-start procedure a large step optimization method, that is, instead of starting with a different initial random solution at each iteration, it is obtained performing a sufficient large perturbation (as our LS-TS procedure we use 3-way interchange), but as short-step we use an step-descent local search with all-pairs neighborhood.

6 Computational experiment

These two procedure described in the previous section was implemented in Delphi, and all experiments were performed on a Pentium 350 MHz personal computer. Before testing the effectiveness of our procedure, we perform several experiments to explore the effect of changes in the search parameters iter.

6.1 Problem generation

For our experiments we use a random generation to generate 40 sequences, each instances was eleven times ran, processing times and setup times was generated using a discrete uniform distribution: $DU(1, 50)$, $proctime = readytime + DU(1, 50)$ due dates was generated as follows: r is a due date factor, p_m a mean processing time, then $duedate = proctime + random(r * p_m)$,the value of r were 0.2 and 0.6, tardiness penalties are drawn from uniform distribution with parameters (1,10). The size of instances varies between 20 and 80 jobs.

Also we present in this section problems instances generated based on the rules of Tan and Narasimhan [15] an also used by Franca in [14]. The size of the instances was 100 jobs. The generation of processing times and setup times follows a discrete uniform distribution: $proctime = readytime + DU(1, 100)$. Due dates are generated according to two parameters: due date range and due date mean. The due date range is defined according to a due date factor, R, and is described by: $dr = r * n * p_m$ where p_m is the mean processing time. The due dates mean is defined by the tardiness factor t, which measures the percentage of jobs that are not expected to accomplish their due dates. The equation for the due dates mean is: $dm = (1-t)*n*p_m$, tardiness penalties are generated as above. We chose values for $t = 0.2$ and $R : 0.4, 0.6$.

6.2 Computational results

Our computational experiments were designed with the goal of comparing the proposed algorithm with the re-start procedure that uses all-pairs neighborhood, by testing our heuristic across a wide range of problems instances.

The analysis of the results were carried out using a non-parametric test (the differences of two medians with hypothesis $H_0 : \varepsilon_1 = \varepsilon_2$, in which ε_1 and ε_2 denote the medians of the two populations of interest), the natural alternative here is $H_1 : \varepsilon_1 < \varepsilon_2$. In our case the value of probability of accepting H_1 when H_0 is true is denoted by α, in our context was assumed the value of $\alpha = 0.05$, for its value $\chi^2 = 3.841$. Then, we calculate the sampling distribution of T, where

$$T = n(|x^o y' - y^o x'| - n)^2/(x^o + y^o)(x' + y')(x^o + x')(y^o + y')$$

where n is the size of the samples arranged together, x^o, y^o denote the values of each sample greater than the median and x', y' the values of each sample less than the median.

In the tables 1 - 6, f_{max}, f_{min}, f_{med}, denote the worst, the better and the median solutions respectively.

The results in the tables 1, 2, 3, 4 and 6 shows that in 53 percent of the instances the hypothesis H_0 was rejected, in 47 percent H_0 was accepted but only in 10 percent of the instance the median of the RS procedure was better than our TS method. In all instances the better solutions were obtained for our TS approach and also in the 46 percent of the instances the worst solutions of TS were betters than the worst solutions obtained for RS.

Table 1

rate	*Pr.*		*TS*			*RS*		*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=20
0.2	1	17123	14698	15873	18369	15276	17445	2.909:3.841
	2	26433	23753	25138	26851	25032	26851	6.545:3.841
	3	30030	27320	28559	32379	28964	30639	1.163:3.841
	4	25781	21734	23714	27681	23115	25286	2.909:3.841
	5	24776	21582	22615	24367	23987	24367	1.163:3.841

Table 2

rate	*Pr.*	*TS*			*RS*			*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=20
0.6	1	29217	26489	27498	29838	28853	29838	1.163:3.841
	2	30213	26875	28108	31292	27740	29252	2.909:3.841
	3	32719	28359	30494	32932	28837	32438	6.545:3.841
	4	23895	19738	21913	24474	22606	23916	6.545:3.841
	5	33869	28966	30467	33838	31024	31877	2.909:3.841

Table 3

rate	*Pr.*	*TS*			*RS*			*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=50
0.2	1	15654.4	13979.9	14915.2	15867.9	15867.9	15867.9	18.187:3.841
	2	17244.9	14632.2	16213.1	17269.8	17269.8	17269.8	18.187:3.841
	3	17848.8	16212.3	17003.8	20658.9	20658.9	20658.9	18.187:3.841
	4	16542.9	14620.1	15212.5	19456.8	17255.9	19456.8	18.187:3.841
	5	17345.2	15265.7	16481.3	19736.1	19736.1	19736.1	18.187:3.841

Table 4

rate	*Pr.*	*TS*			*RS*			*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=50
0.6	1	19408.4	17548.3	18374.4	20313.4	20313.4	20313.4	18.181:3.841
	2	14865.4	13090.5	13817.1	14518.8	14518.8	14518.8	1.543:3.841
	3	17541	15573.5	16809	16606	16606	16606	9.625:3.841
	4	14352.5	12412.3	13528.3	14280.2	14280.2	14280.2	0.000:3.841
	5	19544.9	17989.3	18556.2	20979	20806	20979	18.181:3.841

Table 5

rate	*Pr.*	*TS*			*RS*			*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=80
0.2	1	373228	334833	351536	365485	365485	365485	0.000:3.841
	2	435436	385403	403650	419875	419875	419875	0.550:3.841
	3	402406	358880	377794	415190	393420	415190	11.636:3.841
	4	393559	332563	368696	376273	376273	376273	1.543:3.841
	5	401264	375024	386977	463697	463697	463697	18.181:3.841

Table 6

rate	*Pr.*		*TS*			*RS*		*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=80
0.6	1	420373	379158	399479	395485	395485	395485	7.542:3.841
	2	416704	380567	399659	397624	397624	397624	12.034:3.841
	3	411716	367305	381086	403046	403046	403046	0.550:3.841
	4	362153	317369	32889	352499	352499	352499	0.000:3.841
	5	355184	326653	338028	385665	385665	385665	18.181:3.841

In the second experiment we generate 10 additional instances. In this experiment each instances in RS method was run for 20 minutes. The results in the tables 7, 8 shows that in all instances the hypothesis H_0 was rejected.

-Table 7

rate	*Pr.*		*TS*			*RS*		*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=100
0.4	1	974949	912781	935859	1025791	1025791	1025791	18.181:3.841
	2	1103564	1055710	1074959	1184524	1184524	1184524	18.181:3.841
	3	1047477	9843860	1016378	1062847	1062847	1062847	18.181:3.841
	4	1264833	1193940	1224755	1356399	1356399	1356399	18.181:3.841
	5	1201082	1139007	1175778	1302528	1302528	1302528	18.181:3.841

Table 8

rate	*Pr.*		*TS*			*RS*		*Test:*
r	No	fmax	fmin	fmed	fmax	fmin	fmed	n=100
0.6	1	1097717	1025466	1048813	1216060	1216060	1216060	18.181:3.841
	2	985343	934159	966824	1106013	1106013	1106013	18.181:3.841
	3	785642	742315	763290	923067	885584	923067	18.181:3.841
	4	927086	873667	899510	1016655	1016655	1016655	18.181:3.841
	5	1035347	968024	999947	111662	1112744	111662	18.181:3.841

7 Conclusions

This research prove that tabu search were effective to obtain a good solutions when considering the weighted tardiness objective. The large step optimization seem a good strategy to identifying part of the solution structure and a good solution to initialize the phase of intensification diversification. An additional intensification using path relinking produce goods solution with a low computational cost.

References

[1] M. Laguna and R. Martí (1997) “Grasp and path relinking for 2-layer straight line crossing minimization”, University of Colorado at Boulder.

[2] McNaughton, R. (1959) “Scheduling with deadlines and loss functions”, *Management Science* **6**(1): 1–12.

[3] Held, M.; Karp, R.M. (1962) "A dynamic programming approach to sequencing problems", *Journal of the Society for Industrial and Applied Mathematics* **10**(1): 196–210.

[4] Lawler, E. (1964) "On scheduling problems with deferral costs", *Management Science* **11**(2): 280–288.

[5] Glover, F. (1986) "Future paths for integer programming and links to artificial intelligence", *Computers and Ops. Res.* **5**: 533–549.

[6] Glover, F.; Taillard, E.; de Werra, D. (1993) "A user's guide to tabu search", *Ann. Oper. Res.* **41**: 3–28.

[7] Glover, F.; Laguna, M. (1993) "Modern heuristic techniques for combinatorial problems", published in the Halsted Press, John Wiley & Sons, Inc., chapter 3: 70–147.

[8] Glover, F. (1995) *Tabu Search Fundamentals and Uses.*

[9] Hansen, P. (1986) "The steepest ascent mildest descent heuristic for combinatorial programming", *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.

[10] Ramalhinho-Lourenço, H.; Zwijnenburg, M. (1998) "Combining the large-step optimization with tabu-search: application to the job-shop scheduling problem", in: *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, Ch 14: 219–235.

[11] Emmons, H. (1969) "One-machine sequencing to minimize certain functions of job tardiness", *The Journal of the Operations Research Society of America* **17**(4): 701–715.

[12] Shwimer, J. (1972) "On the N-job, one-machine, sequence-independent scheduling problem with tardiness penalties: a branch-bound solution", *Management Science* **18**(6), B-301.

[13] Morton, T.E.; Pentico, D.W. (1993) "Heuristic scheduling systems, with applications to production systems and project management", Wiley Series in Engineering and Technology Management.

[14] Franca, P.M.; Mendes, A.; Moscato, P. (1999) "A memetic algorithm for the total tardiness single machine scheduling problem", Working Paper, Universidade Estadual de Campinas.

[15] Tan, K.C.; Narasimhan, R (1997) "Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach", *OMEGA, International Journal of Management Science* 2516: 619–634.

[16] Beausoleil, R. (2000) "Intensification and diversification strategies with tabu search: one-machine problem with tardiness objective", in: O. Cairo, L.E. Sucar & F.J. Cantú (Eds.) *Lectures Notes on Artificial Intelligence* 1793, Springer-Verlag: 52–62.

[17] Rubin, P.A.; Ragatz, G.L. (1995) "Scheduling in a sequence dependent setup environment with genetic search", *Computers Ops. Res.* **22**(1): 85–99.