RAIRO Operations Research
RAIRO Oper. Res. **35** (2001) 251-267

EXTENSION OF REVERSE ELIMINATION METHOD THROUGH A DYNAMIC MANAGEMENT OF THE TABU LIST

Saïd Hanafi[1] and Arnaud Fréville[1]

Abstract. The Reverse Elimination Method (REM) is a dynamic strategy for managing the tabu list. It is based on logical interdependencies between the solutions encountered during recent iterations of the search. REM provides both a necessary and sufficient condition to prevent cycling. The purpose of this paper is first to incorporate in REM a *chronological order rule* when cycling is unavoidable, thereby assuring the finite convergence of Tabu Search. Secondly, we correct a generalization of REM, the so-called REM-t method proposed by Glover (1990) where t is an integer parameter which controls the number of tabu attributes. A suitable adjustment of this parameter t can be designed in order to create a balance between diversification and intensification. In this paper, new dynamic rules for controlling the adjustment of the parameter t, are proposed. Finally, to illustrate the differences between the variants proposed for managing the tabu list, we test some of them on the 0–1 multidimensional knapsack problem.

Received May, 1999.

[1] LAMIH, UMR 8530 du CNRS, ROI – Groupe Recherche Opérationnelle et Informatique, Université de Valenciennes et du Hainaut-Cambrésis, Le Mont Houy, 59313 Valenciennes Cedex, France; e-mail: (hanafi,freville)@univ-valenciennes.fr

Résumé. La Méthode d'Élimination Inverse (REM) est une stratégie dynamique pour gérer la liste tabou à chaque itération de la recherche. Elle est basée sur des interdépendances logiques entre les solutions rencontrées pendant les dernières itérations. REM fournit à la fois une condition nécessaire et suffisante pour éviter de cycler. Le but de ce papier est dans un premier temps, d'incorporer dans REM une règle d'ordre chronologique lorsque le cyclage est inévitable, ce qui assure la convergence finie de la recherche tabou. Deuxièmement, nous corrigeons une généralisation de REM proposée par Glover (1990), la méthode REM-t, où t est un paramètre entier contrôlant le nombre d'attributs tabou. D'autre part, un ajustement adéquat de ce paramètre t permet de créer un équilibre entre diversification et intensification. Dans ce papier, de nouvelles règles dynamiques pour contrôler l'ajustement du paramètre t sont proposées. Enfin, quelques-unes des procédures proposées pour gérer la liste tabou sont testées sur le problème du sac-à-dos multidimensionnel en variables $0-1$ afin d'illustrer les différences entre les variantes de la méthode de base.

Keywords: Tabu search, Reverse Elimination Method, 0–1 multidimensional knapsack, strategic oscillation.

1. Introduction

Tabu search (TS) is a metaheuristic introduced by Glover in 1986. It is an improvement neighborhood search approach to overcome local optimality, and it has proved to be highly successful for solving hard combinatorial optimization problems. TS is an iterative method that starts with one initial solution (feasible or infeasible) or a set of initial solutions. Next, the algorithm tries repeatedly to construct from a current solution or a collection of solutions, another solution by searching neighborhoods. The process continues to generate neighboring solutions until a certain stopping criterion is satisfied (a state-of-the art of the method and its applications is given in [10]).

The main components of TS are the introduction of adaptive memory and responsive exploration. Flexible memory is subdivided into short-term memory and long-term memory. The short-term memory also called *recency-based memory* keeps track of the most recent solutions attributes that have been modified during the search processes. The recency-based short memory called *tabu list* (TL) determines moves that are forbidden for a span of time called *tabu-tenure* of tabu list, as a way to control the minimal length of a cycle result during the search. The approach of storing complete solutions generally consumes an enormous amount of space and time when applied to each solution generated. Therefore, instead of storing complete solutions visited, only those attributes which contain mainly information about changes resulting in moving from one solution to another, are recorded in TL.

Frequency-based memory incorporates new dimensions of solution quality and/or move influence based attributes of a subset of the solutions visited. Frequency

measure is decomposed into a transition measure and a residence measure. The transition measure counts the number of times an attribute changes the solutions visited on a particular trajectory. The residence measure counts the number of times an attribute belongs to solutions visited on a particular trajectory or the number of instances where an attribute belongs to solutions from a particular subset. The responsive exploration is based on the supposition that a bad strategic choice can yield more information than a good random choice [6, 7, 10]. In this paper, we focus our intention on Short-term memory functions, which provide the important foundations of the TS methodology that prevents cycling.

Managing a tabu list is a fundamental element of TS. TL can be defined and handled in various ways depending on the problem to be solved. The first is *static*; the attributes of the moves are forbidden for a certain number of iterations, which stays fixed along the search process. This kind of TL management can be implemented by means of a circular list, which eliminates the oldest tabu move at each iteration in order to make room for the new one. The size of TL must be neither too small to avoid cycling and leave unvisited regions, nor too large to prevent the search from blocking.

The second way of managing TL is *dynamic* and depends on the state of the search. It allows us to vary the tenure of the tabu status according to the attributes considered. Some of the dynamic management methods determine a tabu status that is based on sequential relationships between the selected moves to avoid some special cases of cycling. The main strategies are the Cancellation Sequence Method and the *Reverse Elimination Method* (REM) proposed by Glover [7]. Another form of systematic dynamic management consists in subdividing the tabu list into a static part and a dynamic part; this approach is called moving gap [14]. Other adaptive mechanisms have been proposed to adjust the tabu tenure, first for solving combinatorial optimization problems like 0–1 MKP [2], and second for solving the constraint satisfaction problem [17].

Our paper focuses on a generalization of REM introduced by Glover [7], which can be applied to various problems. In the following, we test it just for the 0–1 multidimensional knapsack problem (0–1 MKP) because it provides a convenient basis to illustrate differences between different variants of the basic approach. The 0–1 MKP is a well-known integer programming model, which may be stated as follows:

$$(0{-}1 \text{ MKP}) \left[\begin{array}{lll} \max & \sum_{j=1}^{n} c_j x_j & \\ s.t. & \sum_{j=1}^{n} A_{ij} x_j \leq b_i & i \in I = \{1, ..., m\} \\ & x_j = 0 \quad \text{or} \quad 1 & j \in J = \{1, ..., n\} \end{array} \right.$$

where n is the number of items and m is the number of the knapsack's constraints with capacities b_i, associated weights A_{ij} and profits c_j. The objective is to find a feasible subset of the set of items that yields a maximum profit. The matrix A and the vectors b and c consist of real-valued constants that satisfy $A_{ij} \geq 0$, $b_i > 0$ and $c_j > 0$. The 0–1 MKP has a large domain of applications among which we

can cite resource allocation and capital budgeting [16]. Several exact and heuristic approaches have been developed for solving this NP-hard problem, including tabu search methods in particular [12, 13].

Section 2 reviews the basic ideas of REM using a single attribute and the usual techniques for reducing the number of tracing steps. Section 3 provides a detailed description of a generalization of REM, the so-called REM-t method, and also our main contributions. First, we correct an initial version of REM-t stated in [7] which concerns the update of the running list. Second, we propose dynamic rules for controlling the adjustment of the parameter t, which is a crucial task according to the intensification and diversification strategies. In Section 4, this new dynamic TL management is embedded in the oscillation TS method of Hanafi and Fréville devoted to the 0–1 MKP [12]. Numerical experiments are given in Section 5 and are followed by the conclusion.

2. Reverse Elimination Method

The Reverse Elimination Method is a dynamic strategy for managing the tabu list. Its main principle is the use of logical interdependencies between the solutions encountered during recent iterations of the search. The basic idea of REM is to trace back a *running list* (RL) containing information on all moves performed throughout the search. While backtracing RL, a *residual cancellation sequence* (RCS) is built up, consisting of a sequence of moves or attributes between two solutions where all mutually canceling attributes are eliminated. If the remaining attributes in the RCS can be reversed by exactly one move, then this move is tabu in the next iteration. In case of a single attribute, the tabu moves correspond to RCS reduced to only one attribute.

This initial version of REM is fully described by the following procedure REM (Fig. 1). The symbols "+" and "−" denote the union and difference operators of two sets respectively, and $\overline{e}$ denotes the complement of attribute e. The inputs are the current iteration k and an array RL that contains all attributes moves performed up to iteration k. The outputs are the tabu list TL of tabu-active attributes. Initially the procedure starts with an empty TL and an empty RCS. During the backtracing, RCS is modified successively by either adding or dropping attributes. At the i^{th} tracing step, dropping (respectively adding) an element from (resp. to) RCS increases (resp. decreases) the difference between the current solution x^k visited at iteration k, and the solution x^i previously visited at iteration $i \leq k$.

Example 1. REM using single attribute.

We consider the two simple ADD $[x_j = 0 \rightarrow x_j = 1]$ and DROP $[x_j = 1 \rightarrow x_j = 0]$ moves for solving a 0–1 MKP instance of size $n = 4$. The attribute of the ADD move (respectively DROP) is represented by the index j (resp. $\overline{j}$).

The first two rows of Table 1 represent the running list RL after 11 iterations by indicating the move made at each iteration. The initial solution is (0,0,0,0) and the solution generated at iteration 11 is (0,0,1,0). Following the RL description, each

```
Procedure REM(k,RL,TL)
  Initialization
    TL := ∅;
    RCS := ∅;
  Backtracing
    for i := k to 1 step -1 do begin
        if RL[i]∈ RCS then                          // Restriction of RCS
            RCS := RCS - { RL[i] }
        else                                        // Expansion of RCS
            RCS := RCS + {RL[i]}
        endif ;
        if RCS = {e} then                           // Length of RCS equal to 1
            if ē ∉ TL then TL := TL + { ē } endif
        endif;
    endfor;
```

FIGURE 1. REM using a single attribute.

TABLE 1. Illustration of REM using single attribute.

Running List	Iteration	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
	Move	1	2	3	4	$\overline{3}$	$\overline{2}$	$\overline{1}$	3	2	$\overline{4}$	$\overline{3}$
Tracing step	Residual Cancellation Sequence											
	k = 5	*k* = 6		*k* = 7		*k* = 8		*k* = 9		*k* = 10		*k* = 11
11												**$\overline{3}$**
10											**$\overline{4}$**	$\overline{4}$ $\overline{3}$
9									**2**	2 $\overline{4}$	2 $\overline{4}$	$\overline{3}$
8							**3**	3 2	3 2 $\overline{4}$		2 $\overline{4}$	
7					**$\overline{1}$**	$\overline{1}$ 3	$\overline{1}$ 3 2	$\overline{1}$ 3 2 $\overline{4}$	$\overline{1}$ 3 2 $\overline{4}$	$\overline{1}$ 2 $\overline{4}$		
6			**2**	$\overline{2}$ $\overline{1}$		$\overline{2}$ $\overline{1}$ 3		$\overline{1}$ 3	$\overline{1}$ 3 $\overline{4}$		$\overline{1}$ $\overline{4}$	
5		**$\overline{3}$**	$\overline{3}$ $\overline{2}$	$\overline{3}$ $\overline{2}$ $\overline{1}$		$\overline{2}$ $\overline{1}$		**$\overline{1}$**	$\overline{1}$ $\overline{4}$	$\overline{3}$ $\overline{1}$ $\overline{4}$		
4	4 $\overline{3}$	4 $\overline{3}$ $\overline{2}$	4 $\overline{3}$ $\overline{2}$ $\overline{1}$		4 $\overline{2}$ $\overline{1}$		4 $\overline{1}$		**$\overline{1}$**	$\overline{3}$ $\overline{1}$		
3	**4**	4 $\overline{2}$	4 $\overline{2}$ $\overline{1}$		3 4 $\overline{2}$ $\overline{1}$		3 4 $\overline{1}$		3 $\overline{1}$		**$\overline{1}$**	
2	2 4	**4**	4 $\overline{1}$		3 4 $\overline{1}$		2 3 4 $\overline{1}$		2 3 $\overline{1}$		2 $\overline{1}$	
1	1 2 4	1 4	**4**		3 4		2 3 4		2 3		**2**	

column of Table 1 illustrates the construction of RCS from iteration 5 through 11. The different RCS constructed at iteration $k = 5$ are $\{\overline{3}\}$, $\{4, \overline{3}\}$, $\{4\}$, $\{2, 4\}$ and $\{1, 2, 4\}$ during the tracing steps 5, 4, 3, 2 and 1 respectively.

Clearly, the size of RCS cannot exceed 4, which is the size of the neighborhood of any current solution. For each iteration k, bold indices denote the attributes whose complement will be tabu-active in the next iteration $k + 1$ (at iteration 5, the set of tabu-active attributes is $\{\overline{3}, 4\}$).

```
Procedure REM(k, RL, TL, Last)
  Initialization
    TL := ∅;
    RCS := ∅;
  Backtracing
    for i := k to 1 step -1 do begin
        if RL[i] ∈ RCS then                          // Restriction of RCS
            RCS := RCS - { RL[i] }
        else                                         // Expansion of RCS
            RCS := RCS + {RL[i]}
        endif ;
        if RCS = {e} then                            // Length of RCS equal to 1
            if ē ∉ TL then TL := TL + { ē }; endif
            Last = RCS;
        endif;
    endfor;
```

FIGURE 2. REM with chronological order rule.

2.1. CHRONOLOGICAL ORDER

In Glover [7] and Dammeyer and Voss [3], efficient implementations of the general procedure of REM with useful details are given, especially to avoid cycling. Under the assumption that the moves are reversible and that the attributes satisfy a sufficiency property, the tabu status assigned by REM provides a necessary and sufficient condition to prevent re-visiting solutions already explored. It was noted in [7] that "*situations arise where no move exists that will avoid duplicating a previous solution*". Glover conjectures that a *chronological order rule* for revisiting solutions already encountered (*i.e.* restarting the exploration from the earliest solution visited in the past), has implications for finiteness in 0–1 integer programming and optimal set membership settings. Hanafi [11] has shown that this conjecture is true and other developments related to the convergence of tabu search are given in [8].

In the single attribute case, the initial REM procedure can be easily modified to incorporate the *chronological order rule* when cycling is unavoidable (Fig. 2). If all available moves at iteration k are tabu-active (*i.e.* if all neighbor solutions of the current solution x^k are already visited) then the next solution (iteration $k+1$) selected corresponds to the earliest solution x^i examined during the search. This choice rule is called an *aspiration-by-default* criterion in TS literature. For this purpose a new output parameter *Last* is introduced, which corresponds to the tabu-active attribute inserted last in the current tabu list TL. So, at any iteration for which all the available moves are tabu-active, the next move selected is the complement of the one which is stored in the parameter *Last*. In other words, the indicator *Last* corresponds to the element inserted last in TL.

TABLE 2. REM modified using single attribute.

Running List	Iteration k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	move	1	2	3	4	$\overline{3}$	$\overline{2}$	$\overline{1}$	3	2	$\overline{4}$	$\overline{3}$	4	1	$\overline{4}$	$\overline{2}$	3	$\overline{1}$
Tracing step	Residual Cancellation Sequence																	
	k = 11			k = 12			k = 13			k = 14			k = 15			k = 16			k = 17		
17																		$\mathbf{\overline{1}}$
16														$\mathbf{3}$			3	$\overline{1}$
15										$\mathbf{\overline{2}}$			$\overline{2}$	3		$\overline{2}$	3	$\overline{1}$
14						$\mathbf{\overline{4}}$			$\overline{4}$	$\overline{2}$		$\overline{4}$	$\overline{2}$	3	$\overline{4}$	$\overline{2}$	3	$\overline{1}$
13					$\mathbf{1}$			1	$\overline{4}$	$\overline{2}$	1	$\overline{4}$	$\overline{2}$	3		$\overline{4}$	$\overline{2}$	3
12				$\mathbf{4}$		4	1		1	$\overline{2}$		1	$\overline{2}$	3			$\overline{2}$	3
11			$\mathbf{\overline{3}}$		$\overline{3}$	4	$\overline{3}$	4	1		$\overline{3}$	1	$\overline{2}$					$\mathbf{\overline{2}}$
10		$\overline{4}$	$\overline{3}$			$\mathbf{\overline{3}}$		$\overline{3}$	1		$\overline{4}$	$\overline{3}$	1	$\overline{2}$		$\overline{4}$	1	$\overline{2}$			$\overline{4}$	$\overline{2}$
9	2	$\overline{4}$	$\overline{3}$		2	$\overline{3}$	2	$\overline{3}$	1	2	$\overline{4}$	$\overline{3}$	1			$\overline{4}$	$\overline{3}$	1			$\overline{4}$	1				$\mathbf{\overline{4}}$
8		2	$\overline{4}$			$\mathbf{2}$		2	1		2	$\overline{4}$	1			$\overline{4}$	1		3	$\overline{4}$	1			3	$\overline{4}$
7	$\overline{1}$	2	$\overline{4}$		$\overline{1}$	2			$\mathbf{2}$			2	$\overline{4}$				$\mathbf{\overline{4}}$			3	$\overline{4}$		$\overline{1}$	3	$\overline{4}$
6		$\overline{1}$	$\overline{4}$			$\mathbf{\overline{1}}$			$\varnothing$				$\overline{4}$			$\overline{2}$	3	$\overline{4}$	$\overline{2}$	$\overline{1}$	3	$\overline{4}$
5	$\overline{3}$	$\overline{1}$	$\overline{4}$		$\overline{3}$	$\overline{1}$			$\mathbf{\overline{3}}$			$\overline{3}$	$\overline{4}$			$\overline{3}$	$\overline{2}$	$\overline{4}$		$\overline{2}$	$\overline{4}$		$\overline{2}$	$\overline{1}$	$\overline{4}$
4		$\overline{3}$	$\overline{1}$	4	$\overline{3}$	$\overline{1}$		4	$\overline{3}$				$\mathbf{\overline{3}}$				$\overline{3}$	$\overline{2}$			$\mathbf{\overline{2}}$			$\overline{2}$	$\overline{1}$
3			$\mathbf{\overline{1}}$		4	$\overline{1}$			$\mathbf{4}$				$\varnothing$				$\mathbf{\overline{2}}$			3	$\overline{2}$		3	$\overline{2}$	$\overline{1}$
2		2	$\overline{1}$	2	4	$\overline{1}$		2	4				$\mathbf{2}$				$\varnothing$				$\mathbf{3}$			3	$\overline{1}$
1			$\mathbf{2}$		2	4	1	2	4			1	2				$\mathbf{1}$			1	3				$\mathbf{3}$
Last	2			$\overline{1}$			4			2				1				3				3		

Example 1. (continued)

From the fifth to the eleventh iteration, no more than two moves are tabu-active per iteration, so that a blocking situation arises (see Tab. 1). Such is not the case during the following iterations (Tab. 2). At iteration $k = 12$, 13, 14 and 17, all potential moves are tabu-active. Each time, the search restarts from the earliest solution visited in the past and a cycle is exhibited, a phenomenon marked by an empty RCS in the next iteration $k + 1$. For example, RCS is empty at iteration $k = 13$, step 6, which means that a blocking situation arises at iteration 12, since all the available moves are tabu-active. Therefore the move performed at iteration $k = 13$, corresponds to the complement of $Last = \overline{1}$, then the solution (1,1,0,1), already visited at iteration 6, is revisited at iteration 13.

2.2. Reduction of tracing steps

REM provides both a necessary and sufficient condition to prevent cycling. This observation is valid as long as there is no maximal length of the RL limiting the backtracing. Clearly, the effort required to determine TL by REM increases as the search progresses, since RL is traced at each iteration. Some techniques have been explored to reduce the computational time [3, 7]. Simple strategies consist in decreasing the total number of backtracing or the number of steps in each backtracing and may be made by exact or heuristic rules.

By example, the computational effort of the backtracing is reduced by limiting the number of tracing steps per iteration with a chosen value Max_Step. The parameter Max_Step can be fixed along the search or dynamically adjusted during the search. For example, one way of achieving a dynamic adjustment is to terminate the backtracing after reaching a first local optimum. An alternative way is to decrease the frequency of the backtracing by tracing back the running list only after the constructive and/or destructive phases have been performed (see Sect. 5).

Earlier termination of the backtracing based on a *solo-attribute* are given in [7] and corrected in [3]. A solo-attribute $RL[i]$ is an attribute such that its complement does not appear among earlier entries of the running list. The backtracing terminates as soon as two solo-attributes are found or as soon as the complement of a solo-attribute becomes tabu. This exact rule can be relaxed by stopping the backtracing when only one solo-attribute to become tabu is found. This type of stopping criterion is "spontaneous".

Another stopping criterion, called the "prediction criterion", is based on the observation that at any iteration k, the size of RCS at step i cannot decrease by more than one during the next iteration $k+1$. An array $Least[0...n]$ is then introduced to identify the smallest integer i^* such that backtracing up to step i^* leads to an RCS with a size equal to r (*i.e.* $Least[r] = i^*$). For the 0–1 MKP, the array *Least* strictly identifies at iteration k the position of the first solution encountered (the earliest) whose difference between the current solution x^k is equal to r. The update of *Least* at iteration k is simply made in the REM procedure by setting $Least[r] = k$ for $r = 1, ..., n$ initially and by setting $Least[|RCS|] = i$ at each step i in the backtracing. Generally, it is unnecessary to trace the running list RL during the next p iterations earlier than $i = \min\{Least[r] : r = 0, ..., p+1\}$.

3. Generalization of REM

A simple way to generalize the original version of REM is to increase the minimal set of tabu attributes required to prevent cycling arbitrarily. This may be done by making tabu active the complement of attributes in all RCS whose size is less than or equal to an integer parameter which controls the number of tabu attributes. In this section, firstly we correct a generalization of REM, the so-called REM-t method initially proposed by Glover (1990). REM-t method offers an opportunity to balance between diversification and intensification by a suitable adjustment of the parameter t. Secondly, new dynamic rules for controlling the adjustment of the parameter t, are proposed.

3.1. REM-t procedure

A straightforward implementation of REM-t consists in replacing the instruction:

$$\textbf{if} \quad RCS = \{e\} \quad \textbf{then if} \quad \overline{e} \notin TL \quad \textbf{then} \quad TL := TL + \{\overline{e}\} \quad \textbf{endif}$$

```
Procedure REM-t(k, RL, TL)
  Initialization
    TL := ∅;
    RCS := ∅;
    for e := 1 to q do Min_rep[e] = t + 1;
  Backtracing
    for i := k to 1 step -1 do begin
        if RL[i]‾ ∈ RCS then                                  // Restriction of RCS
            RCS := RCS - { RL[i]‾ }
            Min_rep[RL[i]] = min{Min_rep[RL[i]], |RCS|}
        else                                                  // Expansion of RCS
            RCS := RCS + {RL[i]}
            for (all e ∈ RCS) do Min_rep[e] = min{Min_rep[e], |RCS|}
        endif;
    endfor;
  Tabu Status
    for e := 1 to q do
        if (Min_rep[e] ≤ t and ē ∉ TL) then TL := TL + { ē }; endif.
```

FIGURE 3. Corrected REM-t using a single attribute.

by the following one:

$$\textbf{if} \quad RCS = \{e_1, ..., e_t\} \quad \textbf{then for} \quad k := 1 \quad \textbf{to} \quad t$$
$$\textbf{if} \quad \overline{e}_k \notin TL \quad \textbf{then} \quad TL := TL + \{\overline{e}_k\} \quad \textbf{endfor}; \ \textbf{endif}.$$

But this implementation is very time consuming since the search $\overline{e}_k \notin TL$ can be performed a lot of times for any attribute e_k. To reduce the complexity, Glover introduced in [7] an array Min_$rep[1...q]$ where q is the number of different attributes used during the search. Thus, for the 0–1 MKP using a single attribute, the number of potential attributes is $q = 2n$ but at each iteration, the number of attributes contained in RCS cannot exceed $q^* = n$. The value Min_$rep[e]$ gives the smallest size of RCS containing the attribute e. Therefore, the attribute $\overline{e}$ cannot be tabu if Min_$rep[e] > t$. The procedure REM-t starts with an empty RCS and Min_$rep[e] = t + 1$ for any attribute e. During an expansion step, the attribute $RL[i]$ is added to the current RCS and the value Min_$rep[RL[i]]$ is updated at each tracing step. In Glover's procedure, $\overline{RL[i]}$ is then deleted from the current RCS and only the values of Min_$rep[e]$ related to the predecessors of $\overline{RL[i]}$ are updated during a restricting step i. This is not correct as shown in Example 2. Indeed, the values Min_$rep[e]$ for all the remaining elements e of RCS must be updated. Figure 3 gives a corrected version of the REM-t procedure.

Example 2. Updating of array Min_rep with REM-4.

We consider an instance of the 0–1 MKP with $n = 6$ and a running list RL performed up to iteration $k = 11$. In its first three columns Table 3 gives the knowledge and length of the RCS for each tracing step. The remaining columns

TABLE 3. Update of Min_*rep* with REM-4.

Running List	Iteration k	1	2	3	4	5	6	7	8	9	10	11
	move	4	6	$\overline{3}$	5	$\overline{4}$	$\overline{2}$	4	$\overline{5}$	$\overline{1}$	2	3

Residual Cancellation Sequence							Update of *Min_rep*										
								Corrected version						Glover procedure					
tracing step i	$k = 11$					Length of *RCS*	$e \in RCS$	$\overline{1}$	2	3	4	$\overline{5}$	6	$\overline{1}$	2	3	4	$\overline{5}$	6
							initial	5	5	5	5	5	5	5	5	5	5	5	5
11					3	1	11	5	5	**1**	5	5	5	5	5	**1**	5	5	5
10				2	3	2	10	5	**2**	1	5	5	5	5	**2**	1	5	5	5
9			$\overline{1}$	2	3	3	9	**3**	2	1	5	5	5	**3**	2	1	5	5	5
8		$\overline{5}$	$\overline{1}$	2	3	4	8	3	2	1	5	**4**	5	3	2	1	5	**4**	5
7	4	$\overline{5}$	$\overline{1}$	2	3	5	7	3	2	1	**5**	4	5	3	2	1	**5**	4	5
6		4	$\overline{5}$	$\overline{1}$	3	4	6	3	2	1	**4**	4	5	3	2	1	**4**	4	5
5			$\overline{5}$	$\overline{1}$	3	3	5	3	2	1	4	**3**	5	3	2	1	4	*4*	5
4				$\overline{1}$	3	2	4	**2**	2	1	4	3	5	*3*	2	1	4	*4*	5
3					$\overline{1}$	1	3	**1**	2	1	4	3	5	**1**	2	1	4	*4*	5
2				6	$\overline{1}$	2	2	1	2	1	4	3	***2***	1	2	1	4	*4*	**2**
1			4	6	$\overline{1}$	3	1	1	2	1	**3**	3	2	1	2	1	**3**	*4*	2

compare for $t = 4$ the values of Min_*rep* provided by the procedure of Glover and our corrected version (Min_*rep* is restricted to the six attributes contained in RCS at iteration $k = 11$).

In these columns, the first row indicates the attributes, which appear at least once in the RCS during the tracing steps. The second row shows the initialization of the array Min_*rep* (Min_*rep*$[e] = t + 1 = 5$ for any attribute $e \in RCS$). The following rows record the values of the array Min_*rep* for each tracing step; the elements, which are updated by the REM-4 procedure, are shown in bold.

During the first restriction step ($i = 6$) with Glover's version, Min_*rep*[3] is not updated since 3 is a successor of the eliminated attribute 2 in the previous RCS. Therefore, no values are changed by Glover's version during steps 5 and 4 since the last element in RCS is eliminated. Glover's version is incorrect, since at the end of the procedure ($i = 1$), Min_*rep*$[\overline{5}]$ is equal to 4, which is an error.

3.2. DYNAMIC CONTROL OF THE PARAMETER t

Intensification and diversification are two important components of TS. Intensification strategies are based on recording and exploiting elite solutions or specific features of these solutions. There are many possibilities for modifying choice rules so that they intensify around attractive regions located by elite solutions that have historically been found good. On the other hand, diversification strategies drive the search towards unexplored regions by introducing periodically into the solution attributes that are infrequently used.

Several strategies that create a *discontinuity* in the trajectory of the search have been proposed. Such a discontinuity may be made, for example, by jumping either to a new initial solution for restarting the solution process or to an elite solution.

```
Procedure REM-dynamic (k, RL, TL)
  Initialization
    TL := ∅;
    RCS := ∅;
    for e := 1 to q do Min_rep[e] = k + 1;
  Backtracing
    for i := k to 1 step -1 do begin
        if RL[i] ∈ RCS then                                   // Restriction of RCS
            RCS := RCS - { RL[i] }
            Min_rep[RL[i]] = min{Min_rep[RL[i]], |RCS|}
        else                                                  // Expansion of RCS
            RCS := RCS + {RL[i]}
            for (all e ∈ RCS) do Min_rep[e] = min{Min_rep[e], |RCS|}
        endif;
    endfor;
  Tabu Status
    Choose a rule to compute the parameter t ;
    for e := 1 to q do
        if (Min_rep[e] ≤ t and ē ∉ TL) then TL := TL + { ē }; endif.
```

FIGURE 4. REM-*dynamic* procedure.

Other strategies maintain the *continuity* of the trajectory and provide a balance between intensification and diversification. Such is the case in the oscillation strategy based on alternation between feasible and infeasible domains [9, 12].

REM also creates an opportunity to balance between diversification and intensification by oscillating according to the number of tabu attributes. The adjustment of the parameter t in a REM-t framework can be used to create a balance between intensification and diversification by penalizing or conditionally avoiding the choice of attributes associated with low t values. Suitable values of the parameter are a crucial task. The choice can be made either outside or inside the REM-t procedure. However, an intuitive way is to increase (respectively decrease) the parameter t to enforce diversification (resp. intensification) effect.

The following *REM-dynamic* procedure modifies the previous REM-t procedure by incorporating dynamic rules to control the parameter t based on information recorded by the array Min_*rep* (Fig. 4). Moreover, as the size of RCS cannot exceed $\min\{k, q\}$, Min_*rep* is initialized at an arbitrarily large value fixed at $k+1$. At the end of the backtracing, any attribute e such that Min_*rep*$[e] \neq k+1$, is necessarily contained in a RCS.

Among the several ways by which the information contained in Min_*rep* can be exploited, we propose two rules to compute the parameter t:

$$\begin{array}{llcl} \underline{\textit{Rule 1}} & (\textit{mean value}) & t_k & = \lceil (\Sigma_{e \in I(k)} \text{Min\_}rep[e])/qk \rceil \\ \underline{\textit{Rule 2}} & (\textit{median value}) & t_k & = \lceil (\min\{\text{Min\_}rep[e] : e \in I(k)\} \\ & & & \quad + \max\{\text{Min\_}rep[e] : e \in I(k)\})/2 \rceil \end{array}$$

where k is the number of moves performed up to now, $I(k) = \{e|\text{Min\_}rep[e] \neq k+1\}$, qk is the cardinality of $I(k)$ and $\lceil x \rceil$ denotes the smallest integer greater than or equal to the real value x.

4. TS-oscillation algorithm for the 0–1 MKP

Most of the TS methods devoted to the 0–1 MKP use a search space that differs from the unit cube or the extreme points of the feasible domain. Several attempts have been made such that:

- feasibility is maintained along the process, *i.e.* constraints ($Ax \leq b$) and integrality requirements ($x \in \{0,1\}^n$) are always satisfied [3];
- infeasibility dealing with the integrality requirements is allowed during the process [1, 15];
- infeasibility dealing with the constraint requirements is allowed during the process [2, 9, 12, 13, 15].

According to the numerical results published in the literature, the most promising methods for solving 0–1 MKP seem to be those which are based on *strategic oscillation* [9, 12]. In this section, we investigate the use of REM-t with our *TS-Oscillation* algorithm [12].

Strategic oscillation consists in defining a more or less regular alternating rhythm to cross critical levels in different directions. In the 0–1 MKP case, a critical level is defined as a solution (feasible or infeasible) which is lying on or near the boundary of the feasible domain $\{x|Ax \leq b, x \in \{0,1\}^n\}$. More precisely, critical solutions are included in the subset $\{x|x$ feasible, $\exists j \in J0(x)$ such that $(x+e^j)$ is infeasible$\}$ $\cup$ $\{x|x$ infeasible, $\exists j \in J1(x)$ such that $(x - e^j)$ is feasible$\}$, where $J0(x)$ (respectively $J1(x)$) denotes the subset of components of x fixed at 0 (resp. 1). All these critical solutions constitute the promising zone.

The skeleton of our *TS-Oscillation* algorithm can be stated as follows (Fig. 5). One iteration corresponds to one oscillation (two successive constructive and destructive phases) and this algorithm terminates as soon as a fixed number of iterations has been performed without improving the best feasible solution, or after the maximum number of iterations allowed has been reached.

A crossing exploration is achieved by performing forward paths from the feasible domain toward the infeasible region (*constructive phase*) and backward paths in the opposite direction (*destructive phase*) successively. The constructive (respectively destructive) phase consists in the successive setting of variables equal to 1 (resp. 0) according to specific priority rules, which is specified by the following procedure $TS\_ADD(x, u, TL)$ (resp. $TS\_DROP(x, u, TL)$) described in Figure 6.

The priority rules are the usual ones encountered in well-known greedy algorithms devoted to the 0–1 MKP. In addition, an intensification strategy is activated each time the*promising zone* is reached. Moreover, information deduced from surrogate constraints and the memory structure controls the amplitude of the oscillation. The depth of the oscillations corresponds to the number of times the logical variable *near_feasible* remains unchanged. We have compared two versions

TS-Oscillation Algorithm

step 0: *Initialization*

Choose an initial solution x to be feasible or infeasible;

Initialization of tabu search and oscillation strategic parameters;

Stop := False;

while (Not Stop) **do**

switch (direction) **of**

Constructive Phase : (move from feasible to infeasible)

step C1 : forward to the promising zone;

step C2 : intensification phase around the current solution;

step C3 : diversification phase : moving away from the promising zone to the infeasible domain;

Destructive Phase : (move from infeasible to feasible)

step D1: backward to the promising zone;

step D2: intensification phase around the current solution;

step D3: diversification phase : moving away from the promising zone to the feasible domain;

endswitch

step 7: Update the parameters of tabu search and oscillation strategy;

endwhile

FIGURE 5. TS-Oscillation Algorithm.

Procedure TS_ADD(x, u, TL)	***Procedure TS_DROP(x, u, TL)***
while (x is near_feasible **and** $J0(x) \neq \emptyset$) **do**	**while** (x is near_feasible **and** $J1(x) \neq \emptyset$) **do**
choose j^* := argmax$\{\frac{c_j}{uA^j} \mid j \in J0(x)$ and $j \notin TL\}$	choose j^* := argmin$\{\frac{c_j}{uA^j} \mid j \in J1(x)$ and $j \notin TL\}$
if j^* does not exists **then**	**if** j^* does not exists **then**
Select j^* using the *aspiration-by-default* criterion	Select j^* using the *aspiration-by-default* criterion
endif	**endif**
xj^* := 1;	xj^* := 0;
Update multiplier u and/or TL using REM-*t*;	Update multiplier u and/or TL using REM-*t*;
endwhile	**endwhile**

FIGURE 6. Constructive and destructive phases.

of our TS algorithm that differ from the way the amplitude of oscillations on the infeasible side is controlled.

- *Surrogate constraint* (TS1): the infeasibility of the generated solutions is controlled within a surrogate constraint $uAx \leq ub$ where $u \geq 0$ is an optimal dual multiplier of the LP-relaxation of the 0–1 MKP; then x is *near_feasible* becomes $uAx \leq ub$.
- *Violated constraints permutation* (TS2): the solutions must satisfy at least one constraint $A_s x \leq b_s$ that periodically changes in a deterministic fashion;

if k denotes the number of iterations executed so far, then the condition that x is *near_feasible* becomes $A_s x \leq b_s$ with $s := k$ modulo m.

It is well known that a crucial task in heuristics is the tuning of parameters. The tabu list TL was static in the initial implementation of the *TS-Oscillation* algorithm and we obtained good results over a well-known benchmark of test problems described in the literature ([12], Sect. 5). However, these attractive results required quite some processing time in order to determine the suitable size of the tabu list for each instance. As discussed in the previous section, the dynamic management of the tabu list *via* the generalized Reverse Elimination Method provides a strategy to overcome this difficulty. The main drawback is to deteriorate the quality of the solutions, but the robustness of the method will be proved if the quality of the best generated feasible solution remains within a reasonable range. A slight modification of the *TS-Oscillation* algorithm is necessary, corresponding to the input parameter TL of the *TS_ADD(x, u, TL)* and *TS_DROP(x ,u, TL)* procedures, now controlled by the *REM-dynamic* procedure.

5. Numerical experiments

In this section, we present computational results with different variants of the *TS-Oscillation* algorithm coupled with the *REM-dynamic* procedure to manage the tabu list. All the procedures have been coded using C language. The runs are performed on a ultra-sparc SUN station. All running times are given in seconds (CPU time). The maximum number of iterations is at most $10n$.

The robustness of our *TS-Oscillation* algorithm with a dynamic management of the tabu list was tested on two sets of instances of 0–1 MKP. The first one is followed from Fréville and Plateau [4,5] and is composed of 54 instances. Optimal solutions are known for all instances. The second set is due to Glover and Kochenberger [9] and is composed of 24 instances. All the best feasible solutions provided by our dynamic *TS-Oscillation* algorithm are compared with the optimal solution or the best known feasible solution ($\nu(P)$ in both cases).

Table 4 reports numerical comparisons related to the relative error $gap = \frac{(\nu(P)-\nu(TS_i))}{\nu(P)}$ and the CPU time. The following variants of our *TS-Oscillation* algorithm depend on the two main parameters described above:

- the dynamic strategy of the tabu list: REM-1, REM-2 or REM-*dynamic*;
- the control of the oscillations: TS1 or TS2.

The results show that the *TS-Oscillation* algorithm coupled with a dynamic management of the tabu list provides a robust strategy since the relative error never exceeds 4.48%. When the backtracing is made after each move, TS2 generally obtains better results in terms of quality than TS1 but in up to five times more running time. REM-*dynamic* is the best variant in terms of solution quality; however, its running time is slightly greater than that of the two other methods, REM-1 and REM-2.

TABLE 4. *TS-Oscillation* algorithm with a dynamic management of the tabu list.

	REM-1				**REM-2**				**REM-*dynamic***			
	CPU	#	mean	max	CPU	#	mean	max	CPU	#	mean	max
TS1	1.58	10	0.67	4.48	2.19	10	0.67	4.48	2.28	7	0.65	4.48
TS2	7.27	12	0.65	4.48	11.65	12	0.65	4.48	13.64	9	0.59	3.32

CPU: backtracing after each move #: the number of instances for which the gap = 0
mean: the gap average over the 78 instances max: the greatest gap

TABLE 5. Backtracing after each 1/2 oscillation.

	REM-1				**REM-2**			
	CPU	#	mean	max	CPU	#	mean	max
TS1	1.11	11	0.68	4.48	1.44	5	0.78	4.48
TS2	2.06	11	0.60	4.48	3.15	8	0.68	4.48

More detailed results related to the interdependencies between the two main parameters (TL management; oscillation control) are given in the following figures. The performances of REM-2 and REM-*dynamic* coupled with the strategic oscillation TS2 are compared with a selected subset of test problems in Figure 7 (gap) and Figure 8 (CPU time).

The impact of the frequency of the backtracing is shown in Figure 9. When the backtracing is made after 1/2 oscillation, the above comparisons remain valid. However, it is not surprising to observe a decrease of the running time by almost four times as well as a small degradation of the solution quality (see Tab. 5).

The effect of reducing the number of tracing steps has also been measured. In all cases, the earlier termination of the backtracing based on a solo-attribute allows us to reduce the running time significantly, particularly for large-size problems (greater than 100 variables). In case of small-size instances (less than 100 variables), REM procedure with the prediction criterion using the array *Least* needs more CPU time than the REM version without reduction. However, this prediction criterion becomes interesting for large-size problems.

6. CONCLUSION

This paper investigates the dynamic management of the tabu list with the Reverse Elimination Method. First, two technical improvements are provided: a chronological order rule is introduced in REM to prevent cycling and the initial version of REM-t given by Glover is corrected. Second, we propose a new REM-*dynamic* procedure which modifies REM-t by controlling the parameter t

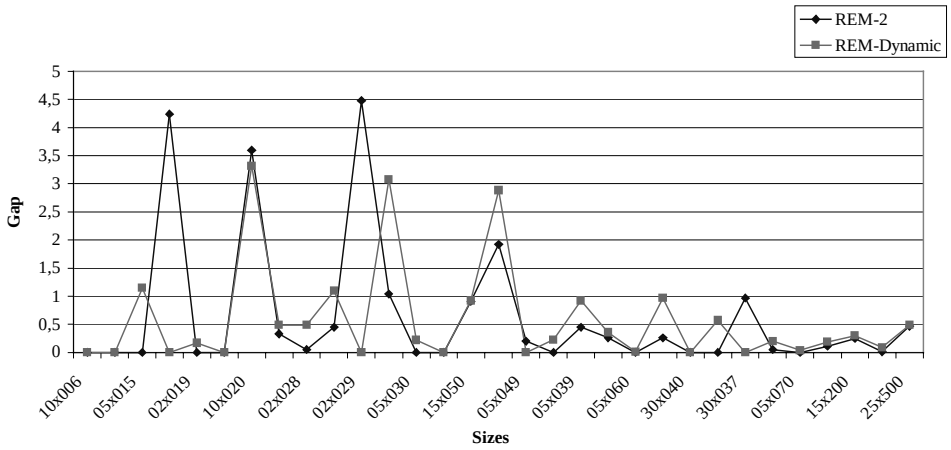

FIGURE 7. Comparison of REM-2 and REM-*dynamic*.

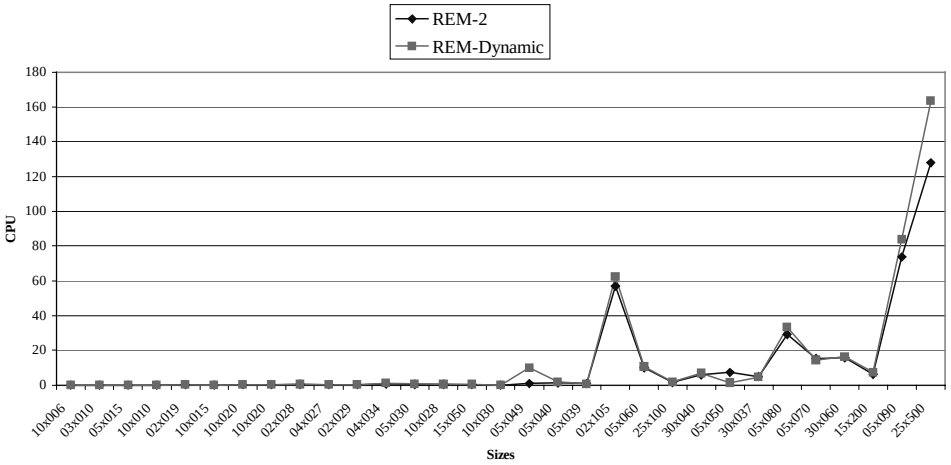

FIGURE 8. Comparison of REM-2 and REM-*dynamic*.

dynamically. REM-t has been validated on the 0–1 multidimensional knapsack problem in order to show the different approaches proposed in this paper to manage the tabu list. Note that all these approaches can be applied to numerous other problems.

REFERENCES

[1] R. Aboudi and K. Jörnsten, Tabu Search for General Zero-One Integer Programs Using the Pivot and Complement Heuristic. *ORSA J. Comput.* **6** (1994) 82-93.
[2] R. Battiti and G. Tecchiolli, The Reactive Tabu Search. *ORSA J. Comput.* **6** (1994) 126-140.

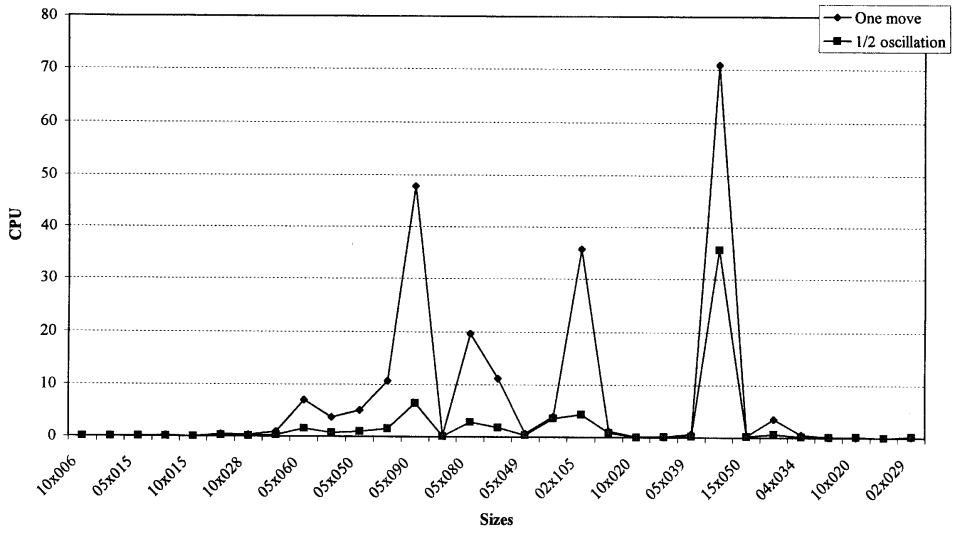

FIGURE 9. Frequency of the backtracing.

[3] F. Dammeyer and S. Voss, Dynamic Tabu List Management using the Reverse Elimination Method. *Ann. Oper. Res.* **41** (1993) 31-46.
[4] A. Fréville and G. Plateau, Heuristics and Reduction Methods for Multiple Constraints 0–1 Linear Programming Problems. *European J. Oper. Res.* **24** (1986) 206-215.
[5] A. Fréville and G. Plateau, Hard 0–1 Multiknapsack Test Problems for Size Reduction Methods. *Investigacion Oper.* **1** (1990) 251-270.
[6] F. Glover, Tabu Search, Part 1. *ORSA J. Comput.* **1** (1989) 190-206.
[7] F. Glover, Tabu Search, Part 2. *ORSA J. Comput.* **2** (1990) 4-32.
[8] F. Glover and S. Hanafi, *Tabu Search and Finite Convergence*, Working paper, LAMIH-UMR CNRS N° 8530. University of Valenciennes, France, presented at INFORMS Meeting, 25-28 October 1998, USA.
[9] F. Glover and G.A. Kochenberger, Critical Events Tabu Search for Multidimensional Knapsack Problems, in *Metaheuristics: Theory and Applications*, edited by I.H. Osman and J.P. Kelly. Kluwer (1996) 407-428.
[10] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic Publishers (1997).
[11] S. Hanafi, On the Convergence of Tabu Search. *J. Heuristics* (to appear).
[12] S. Hanafi and A. Fréville, An efficient Tabu Search Approach for the 0-1 Multidimensional Knapsack Problem. *European J. Oper. Res.* **106** (1998) 659-675.
[13] S. Hanafi, A. Fréville and A. El Abdellaoui, Comparison of Heuristics for the 0–1 Multidimensional Knapsack Problem, in *Metaheuristics: Theory and Applications*, edited by I.H. Osman and J.P. Kelly. Kluwer (1996) 449-466.
[14] R. Hübscher and F. Glover, Applying Tabu Search with influential diversification to multiprocessor scheduling. *Comput. Oper. Res.* **13** (1994) 877-884.
[15] A. Lokketangen and F. Glover, Probabilistic Move Selection in Tabu Search for Zero-One Mixed Integer Programming Problems, in *Metaheuristics: Theory and Applications*, edited by I.H. Osman and J.P. Kelly. Kluwer (1996) 467-488.
[16] J. Lorie and L.J. Savage, Three Problems in Capital Rationing. *J. Bus.* **28** (1955) 229-239.
[17] K. Nonobe and T. Ibaraki, A Tabu Search Approach to the CSP (Constraint Satisfaction Problem) as a General Problem Solver. *European J. Oper. Res.* **106** (1998) 599-623.