ELSEVIER

European Journal of Operational Research 106 (1998) 659-675

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

An efficient tabu search approach for the 0-1 multidimensional knapsack problem

Saïd Hanafi [1], Arnaud Freville *

Université de Valenciennes, LIMAV, Le Mont Houy, B.P. 311, 59304 Valenciennes Cedex, France

Received 1 April 1996; received in revised form 1 April 1997

Abstract

In this paper, we describe a new approach to tabu search (TS) based on strategic oscillation and surrogate constraint information that provides a balance between intensification and diversification strategies. New rules needed to control the oscillation process are given for the 0-1 multidimensional knapsack (0-1 MKP). Based on a portfolio of test problems from the literature, our method obtains solutions whose quality is at least as good as the best solutions obtained by previous methods, especially with large scale instances. These encouraging results confirm the efficiency of the tunneling concept coupled with surrogate information when resource constraints are present.

Keywords: Tabu search; 0-1 Multidimensional knapsack; Strategic oscillation; Surrogate duality

1. Introduction

The 0-1 multidimensional knapsack problem (0-1 MKP) is a generalization of the 0-1 knapsack problem and a special case of general 0-1 integer programming. Formally, the 0-1 MKP problem may be defined as follows:

(0-1 MKP)

$$\max \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} A_i^j x_j \leqslant b_i, \quad i \in I = \{1, \ldots, m\},$$

$$x_j \in \{0,1\}, \quad j \in J = \{1, \ldots, n\},$$

where n is a number of items and m is a number of knapsack's constraints with capacities $b_i (i \in I)$, associated weights $A_i^j (i \in I;\ j \in J)$ and the profits $c_j (j \in J)$. 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 in real-valued constants that satisfy $A \geqslant 0$, $b > 0$ and $c > 0$.

* Corresponding author. Fax: 33 27 14 11 89; e-mail: freville@univ-valenciennes.fr.

[1] E-mail: hanafi@univ-valenciennes.fr.

0377-2217/98/$19.00
PII S0377-2217(97)00296-8

The large domain of applications of this NP-hard problem has greatly contributed to its fame. Among these, we can cite the areas of capital budgeting and resource allocation, the paper of Lorie and Savage being probably one of the earliest reference [49]. Other applications include project selection [53], cutting stock problems [27], loading problems [6,55] determining the optimal investment policy for the tourism sector of a developing country [26], and more recently, delivery of groceries in vehicles with multiple compartments [7] and approval voting [56]. It also appears in the collapsing problem [14] and as a subproblem in large models for allocating processors and data in distributed computer systems [23,24]. Finally, the 0-1 MKP model is frequently used as a benchmark problem to compare or validate general purpose methods in the field of combinatorial optimization.

As for any hard optimization problem, like the 0-1 MKP, a reasonable effort to cope with the problem consists of trying to derive heuristics which solve it suboptimally and which, possibly, yield a good trade-off between the solution quality and the time and the memory requirements. The main drawback of algorithms such as greedy or deterministic exchange algorithms is their inability to continue the search upon becoming trapped in a local optimum. Metaheuristics perform different search strategies to overcome local optimality [52]. The most popular methods are Simulated Annealing (SA) [10,45,50], Tabu Search (TS) [30,31,42], Noising Method (NM) [8], Threshold Accepting Algorithms (TA) [11,12], Neural Networks (NN) [5,44,51], Genetic Algorithms (GA) [38,43,57] and Greedy Randomized Adaptive Search Procedure (GRASP) [15]. Several heuristics or metaheuristics have been used to solve the 0-1 MKP, an updated and comprehensive survey for 0-1 MKP dealing with applications, complexity and heuristics can be found in Fréville [16,18].

In this paper, we propose a new approach of TS based on the strategic oscillation and surrogate constraint information that gives a balance between intensification and diversification strategies. Our method, strongly related to the work of Glover and Kochenberger [34] differs however in several points. Both methods are based on flexible memory, subdivided into short and long term memory by incorporating recency-based memory and frequency-based memory, respectively, but their implementations are quite different. New rules needed to control the oscillation process are given for the 0-1 MKP. Applied to a portfolio of test problems from the literature, our method obtains solutions whose quality is at least as good as the best known solutions obtained by previous methods, especially with large scale instances.

A classification of TS approaches adapted to solve 0-1 MKP is presented in Section 2. Section 3 describes the skeleton of our approach. Section 4 defines the concept of the *promising zone* and the moves to reach this region. Section 5 focuses on the diversification and intensification strategies, and TS and oscillation strategy parameters are fully described in Section 6. Computational results are given in Section 7 followed by the conclusions.

2. Review of tabu search approaches for the 0-1 MKP

The term *Tabu Search* was first coined by Glover [30,31,33] from whom it has derived its modern form. Seminal ideas of the method are also developed by Hansen [42] in a steepest ascent/mildest descent formulation. This method is certainly one of the most popular metaheuristics and has enjoyed successes in a great variety of problem settings (e.g. Glover and Laguna [35]).

The main components of TS are the introduction of adaptive memory and responsive exploration as a basis for solving combinatorial and nonlinear problem. Flexible memory is subdivided into short and long term memory by incorporating recency-based memory and frequency-based memory. The responsive exploration is based on the supposition that a bad strategic choice can yield more information than a good random choice.

TS approaches devoted to 0-1 MKP can be categorized with regard to constraints ($Ax \leqslant b$) and integrality ($x \in \{0,1\}^n$) requirements:

Feasibility is maintained along the process, i.e. constraints and integrality requirements are always satisfied: Dammeyer and Voss [9] developed a TS version for solving 0-1 MKP, based on a proposal

by Glover [31] called the *Reverse Elimination Method* (REM). The tabu list is managed in a dynamic way and a multiattribute move, the so-called *drop_add* move, is used in the solution space $\{0,1\}^n$ which maintains constraints feasibility ($Ax \leqslant b$). The authors show in particular that their TS approach using REM outperforms SA with respect to various criteria.

Infeasibility dealing with the integrality requirements is allowed during the process: Other TS approaches have been designed with the ability to solve zero-one mixed integer programming (0-1 MIP) which are more general models than the 0-1 MKP. All these methods relaxed the integer requirements ($x \in \{0,1\}^n$). Aboudi and Jörnsten [1] superimposed a TS framework on the Pivot and Complement heuristic by Balas and Martin [2] used as a black box, while Lokketangen et al. [48] embedded TS within the Pivot and Complement heuristic.

A more direct approach is used in Glover and Lokketangen [36], using a standard bounded variable simplex method as a subroutine, to exploit the fact that an optimal solution to the 0-1 MIP problem may be found at an extreme point of the LP feasible set. Infeasible solutions are allowed and controlled with special rules expressed as functions of integer infeasibility measures and objective function values. Glover and Lokketangen [37,47] extended their previous work applying first-level TS mechanisms for 0-1 MKP by including strategic oscillation, diversification schemes, probabilistic measures for move selection, and also target analysis to identify effective search parameters and choice rules.

Infeasibility dealing with the constraint requirements is allowed during the process: Battiti and Tecchiolli [4] proposed a TS method, called *Reactive Tabu Search* (RTS), where the move is a simple exchange of one component of the binary current solution. RTS defines the dynamic way by which the tabu list is managed. A penalty factor is introduced in the objective value to overcome the difficulty of satisfying resource requirement ($Ax \leqslant b$) and to transform 0-1 MKP into an unconstrained problem. Generally this penalty factor is proportional to the degree of violation of the constraints and favors feasible solutions. The reported computational experiments with RTS demonstrated an effective performance compared with GA, NN and SA.

Recent advances summarized in [33] allow new mechanisms for dealing with infeasibility, which make use of a *tunneling effect*. Glover and Kochenberger [34] obtained computational results of high quality with a new TS approach employing a flexible memory structure that integrates recency and frequency information keyed to "critical events" of the search process. Their TS method consists of a strategic oscillation scheme which alternates between "constructive" and "destructive" phases and drives the search to variable depths on each side of the feasibility boundary. Hanafi et al. [17] also introduced an associated TS strategy by considering the following modification of the *drop_add* move as described below in *infeasible_drop_add* move. Various mechanisms exist to control the infeasibility of the generated solutions such as 'supplementary resources', 'surrogate constraints' and 'violated constraint permutation'. As in [34,47], optimal solutions are found by Hanifi et al. [17] for each of the 54 test problems reported in [17,19].

Move infeasible_drop_add(x)
{*infeasible_drop_add* defines a move from a feasible solution x to a neighbor of x which is not necessarily feasible}
{step DROP}
choose $j^* = \text{argmin}\{\frac{c_j}{A_j^i} \mid x_j = 1, j \in J\}$ with i^* being a scarce resource such that

$$i^* = \text{argmin}\left\{\left[b_i - \sum_{j=1}^{n} A_i^j x_j\right] \Big/ b_i \mid i \in I\right\};$$

$x_j^* := 0$;
{step ADD}
near_feasible := true;
while near_feasible **do**
choose
$k^* = \text{argmax}\{c_j \mid x_j = 0, j \in J,\ j \neq j^*\}$
with $x + e^{k^*}$ satisfying near-feasible requirements ($x + e^{k^*}$ and x are identical solutions except the k^*-component fixed at 1 in $x + e^{k^*}$);
if k^* exists **then** $x_{k^*} := 1$ **else** near_feasible := false **endif**
endwhile

3. Tabu search based on oscillation strategy

In TS methods, intensification forces the search to examine attractive regions while diversification drives the search into new unexplored regions. The main goal is then to find an efficient balance between intensification and diversification strategies. In this section, we propose a TS approach to achieve a balance based on strategic oscillation and generalized greedy algorithms guided by surrogate constraint information and the of the search. This is, in principle, the approach embodied in the paper by Glover and Kochenberger [34].

Strategic oscillation consists in defining a more or less regular alternating rhythm to cross critical levels in different directions. Critical levels depend on the problem setting, for example a particular form of graph's structure, a subset of the feasible domain, a target value, balance states, local optima, and so on. In the 0-1 MKP case, the critical levels considered in [34] are feasible solutions lying on or near the boundary of the feasible domain. These critical solutions belong to the subset $\{x \mid x$ feasible, $\exists j \in J0(x)$ such that $(Ax + e^j)$ infeasible$\}$. In our approach, infeasible solutions lying on or near the boundary of the feasible domain are also considered as critical solutions according to the oscillation strategy. All these critical solutions constitute the *promising zone*. This zone is included in the subset $\{x \mid x$ feasible, $\exists j \in j0(x)$ such that $(x + e^j)$ infeasible $\} \cup \{x \mid x$ infeasible, $\exists j \in j1(x)$ such that $(x - e^j)$ feasible$\}$, where $J0(x)$ (respectively $J1(x)$) denotes the subset of components of x fixed at 0 (resp. 1).

Crossing and intensive exploration of the promising zone is controlled by using information deduced from the surrogate constraints and the memory of the search. A crossing exploration is achieved by introducing forward paths from the feasible domain toward the infeasible region, and introducing backward paths in the opposite direction. These paths are built, respectively, by performing constructive and destructive phases. An intensification strategy is then activated each time the promising zone is reached. The skeleton of our TS method can be stated as follows.

TS_*Oscillation Algorithm*
{step 0: Initialization}
- Choose an initial solution x to be feasible or infeasible
- Initialization of tabu search and oscillation strategy parameters
- stop := False

while not stop **do**
switch direction **of**
Constructive Phase: (move from a feasible solution to an infeasible solution)
{step C1} Forward to the promising zone
{step C2} Intensification phase around the current solution
{step C3} Diversification phase: moving from the promising zone to the infeasible domain
Destructive Phase: (move from an infeasible solution to a feasible solution)
{step D1} Backward to the promising zone
{step D2} Intensification phase around the current solution
{step D3} Diversification phase: moving from the promising zone to the feasible domain
endswitch
{step U} Update the tabu search and oscillation strategy parameters
end while

An intuitive geometric interpretation of the algorithm above is provided in Fig. 1, where each iteration corresponds to a path linking two solutions, crossing the promising zone. This path is built through a constructive phase (steps Ci, $i = 1, 2, 3$) and a destructive phase (steps Di, $i = 1, 2, 3$), driving the search on each side of the feasibility boundary to detect feasible solutions of high quality. All the steps are described in fuller detail in the following sections.

This algorithm terminates as soon as a selected number of iterations has been performed without improving the best feasible solution, or after a total iteration limit has been reached (for example a fixed number of constructive/destructive phase executions).

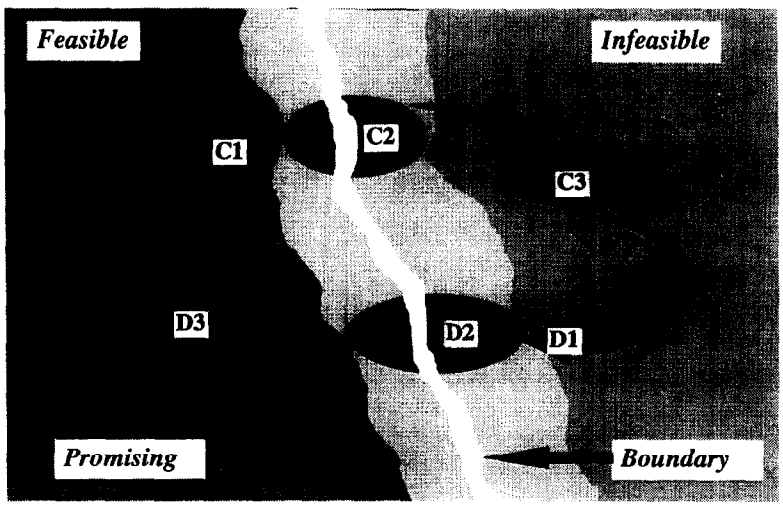

Fig. 1. A geometric interpretation of TS_Oscillation algorithm.

The implementation of this approach allows great flexibility in choosing the crossing procedures, the TS strategy, the boundary definition, the oscillation's amplitude, and the regulation of the change of direction, and so forth. In the remainder of this paper, we propose general procedures to implement the different steps Ci and Di for $i=1, 2, 3$ and to control and update the TS and oscillation strategy parameters.

All the procedures described in the following sections use some common elements. First, the elementary moves in our method are built with two simple exchange moves: ADD(x,j) that sets the component $j \in J0(x)$ at 1 and DROP (x,j) that sets the component $j \in J1(x)$ at 0. Second, the tabu list (TL) parameter contains the attribute of tabu moves. It is well known that the way of modifying this TL parameter, as the search progresses, is an important design characteristic of a TS strategy. In all the procedures the TL parameter's management has been omitted but it can be easily incorporated, once the TS strategy is defined. Finally, the surrogate multiplier u ($u \geqslant 0$ and $uAx \leqslant ub$) can be derived in various ways as discussed in Section 6.

4. Forward and backward to the promising zone

In this section, we present generalized greedy moves used to explore the promising zone. The most well-known greedy algorithms used for the 0-1 MKP are based on constructive or destructive phases. The constructive (resp. destructive) phase consists in setting successively variables equal to 1 (resp. 0) according to specific priority rules. These priority rules use information such as a multiplier $u \geqslant 0$ or a surrogate constraint $uAx \leqslant ub$ to guide the search. The nonnegative multiplier u can also be derived by direct calculations using structural information provided by the entries and the current solution (see Section 6).

The procedure TS_ADD(x,u,TL) generalizes the most well-known greedy algorithm used for the knapsack problem. It tries to improve a current feasible solution x by fixing at 1 variables according to decreasing order of the c_j/uA^j ratios, as far as possible. TS_ADD(x,u,TL) moves can be viewed as a constructive phase defining a path from a feasible solution x to a better solution lying in the promising zone (step C1).

Procedure TS_ADD(x,u,TL)
  feasible := true;
  while feasible **do**
    choose j^* := argmax $\{\frac{c_j}{uA^j} | x_j = 0,$
    $Ax + A^j \leqslant b$ and $j \notin$ TL$\}$;
    if j^* exists **then** ADD(x,j^*) **else** feasible := false **endif**
  endwhile

An alternative way to reach this promising zone from an infeasible solution (step D1) is to project this solution onto the feasible domain as in [54]. This goal is achieved with TS_PROJECT(x,u, TL) moves which builds a path from the infeasible solution x to a feasible one lying in the promising zone.

Procedure TS_PROJECT(x,u,TL)
  feasible := false;
  while not feasible **do**
  choose j^* := argmin$\{\frac{c_j}{uA^j} \mid x_j = 1$ and $j \neq \in$ TL$\}$;
    DROP(x,j^*);
    if $Ax \leqslant b$ **then** feasible := true **endif**
  endwhile

In the constructive procedure TS_ADD and destructive procedure TS_PROJECT, the tabu status of a move can be overruled if certain conditions are met. These conditions, called *aspiration criteria*, are based on the solutions' thresholds of attractiveness. The default aspiration criterion most frequently used is to forbid the tabu status of moves if they lead to a new feasible solution better than the current best. This is the first criterion, particularly applied when the promising zone is reached, we have used. It can be implemented for example, calling the procedures with the parameter TL set to $\emptyset$ (empty set) and the new solution obtained is compared with the best solution found so far.

5. Intensification and diversification

This section focuses on two important components of TS: intensification and diversification.

5.1. Intensification

Intensification aims at concentrating the search in the promising zone. Several numerical experiments in the literature have shown the usefulness of performing a complementing phase to improve the current feasible solution [2,34,46]. The following COMPLEMENT(x,u,TL,V) procedure defines a kind of intensification phase which tries to improve a solution x in the promising zone by exploring alternative solutions in a neighborhood of x. The structure of this neighborhood is controlled with a subset V of components of x, i.e. $V \subseteq J$, which is a parameter of the COMPLEMENT procedure given below. Time complexity requirements restrict our choice to start the search from neighbors x' with a Hamming distance from x equal to 1, i.e. $\{x' \in \{0,1\}^n \sum_{j \in V} |x'_j - x_j| = 1\}$.

Procedure COMPLEMENT(x,u,TL V)
  $x' := x$;
  for all $j \in V$ **do**
    $x'' := x'$;
    if $j \in J0(x')$ **then**
      ADD(x'', j); TS_PROJECT(x'', u, TL)
    else $\{j \in J1(x')\}$
      DROP(x'', j); TS_ADD (x'', u, TL)
    endif
    if $cx'' > cx$ **then** $x := x''$ **endif**
  endfor

This procedure specifies the steps C2 and D2, that correspond to the intensification phases around a critical level lying in the promising zone.

5.2. Diversification

The TS diversification strategy consists in driving the search into new regions. In our approach, the new regions will be reached by moving away from the promising zone toward the feasible or infeasible domain.

We give the procedures to oscillate on each side of the promising zone as follows. The TS_DROP move starts from the promising zone and progresses toward the interior of the feasible domain, with goal of feasible domain is controlled with a *depth* parameter, specifying the maximum number of DROP moves allowed as described below in TS_DROP.

Procedure TS_DROP (x,u,TL,*depth*)
  count = 1;
  while (count $\leqslant$ depth) and ($J1(x) \neq \emptyset$) **do**
    choose j^* := argmin $\{\frac{c_j}{uA^j} \mid x_j = 1$ and

$j \neq \in$ TL};
if not (j^* exists) **then** use aspiration criterion **endif**
DROP (x,j^*);
count = count + 1;
endwhile

The second aspiration criterion is used in the TS_DROP procedure when all the components fixed at 1 are tabu. In this case, the tabu status is overruled to choose j^* by the following rule:

$$j^* := \text{argmin}\left\{ \frac{cx - c_j}{\sum_{i=1}^{m}(b_i - A_i x + A_i^j)} : \right.$$
$$\left. j \in \text{LT and } x_j = 1 \right\}.$$

This rule means that the selected variable reduces the current cost value the least amount possible and at the same time releases the greatest amount of resources possible (i.e., the rule balances these goals by the ratio calculation).

In a symmetrical way, the TS_INFEASIBLE_ADD(x,u,TL) procedure switches the search towards the infeasible region (step C3). A local variable S of type set forces the exploration of all the components of x set to zero and avoid' cycling. For example, consider the following instance described in Table 1, where: $m = 1, n = 5, u = 1, b = 10$ where the logical variable *near_feasible* receives the value "true" when the current solution x is feasible $Ax \leqslant b$.

Therefore, without the set S, the call of the procedure TS_INFEASIBLE_ADD(x,u,TL) does not change the feasible solution x and an infinite loop occurs. By using the set S, the procedure improves the initial solution x by forming a new solution x^*.

Table 1

j	1	2	3	4	5
c	8	13	6	1	7
A	2	5	3	1	7
TL			T		T
x	1	0	0	0	1
x^*	1	0	0	1	1

Procedure TS_INFEASIBLE_ADD (x,u,TL)
$S := \emptyset$;
near_feasible := true;
while near_feasible **do**
choose j^* := argmax $\{\frac{c_j}{u_i A^j} \mid x_j = 0, j \notin \text{TL}$ and $j \notin S\}$
if j^* exists **then**
$x' := x$; Add (x', j^*);
if (x' is near feasible) **then** $x := x'$ **else** $S := S \cup \{j^*\}$ **endif**
else near_feasible := false **endif**
endwhile

The number of times the logical variable *near_feasible* remains unchanged controls the amplitude of the oscillation into the infeasible domain. The infeasibility of the generated solutions allowed during the process can be defined in variety of ways by using surrogate duality information (Sections 6.1 and 6.2).

6. Controlling tabu search and oscillation parameters

The flexibility which allows metaheuristics to apply to a large field of applications is accompanied by the requirement of finding appropriate parameters for specific problems. This is often quite difficult and time consuming [3]. Indeed, selecting these parameters requires either a deep knowledge of the problem structure or a long self-tuning process that is not always reusable. Different strategies are employed to try to overcome this difficulty. Learning, target analysis and case-base reasoning [33,37,39], and probabilistic and parallelization methods [47,58], are among the promising approaches. In this section, we describe how we control the depth of the oscillation and manage the memory of our TS. To achieve this goal, surrogate duality is fully used in different stages of the process.

6.1. Use of surrogate duality information

Surrogate relaxation is another powerful concept used to provide approximate solutions. The

seminal idea is due to Glover [28] and has been developed in several papers (see, for example, [29]). It consists in capturing valuable problem information, not contained in the original constraints considered one at a time, by forming nonnegative linear combinations of the constraints (where the source constraints can include cutting planes).

As with Lagrangian heuristics, the infeasible solutions generated during the process of solving the surrogate dual can be used as starting points in a local search to reach promising feasible solutions. Another approach investigated in this work is to exploit information provided by a surrogate constraint $uAx \leqslant ub$ associated with a multiplier $u \geqslant 0$. The nonnegative multiplier u can be derived in several different ways. It may be the optimal solution of any dual problem associated with the primal problem (P); several attractive methods have been developed based on this idea and coupled with a surrogate dual solver [21,22,25]. The nonnegative multiplier u can also be derived by direct calculations involving structural information provided by the entries and the current solution.

The rules used in our procedures are the following, where $\delta(x)$ (sometimes denoted simply by δ) identifies the slack vector, $\delta(x) = Ax - b$, associated with the current solution x. In the TS_ADD and TS_INFEASIBLE_ADD procedures, the ratio c_j/uA^j is replaced by the cost c_j of item j, i.e. the index j^* of the variable to set at 1, is chosen as follows:

$$j^* := \text{argmax}\{c_j \mid x_j = 0, j \notin \text{TL and } j \notin S\}.$$

In the procedure TS_PROJECT, we use the same multiplier as selected by Senju and Toyoda [54], i.e.

$$u_i := \begin{cases} -\delta_i & \text{if } \delta_i < 0, \\ 0 & \text{otherwise.} \end{cases}$$

For the procedure TS_DROP, we use the multiplier proposed by Fréville and Plateau [17] and Dammeyer and Voss [9]:

$$u_i := \begin{cases} 1 & \text{if } i = \text{argmin } \{\frac{\delta k}{b_k} \mid k = 1, \ldots, m\}, \\ 0 & \text{otherwise.} \end{cases}$$

In the intensification phase, the COMPLEMENT procedure is called with a multiplier that approximates the optimum of the surrogate dual (SD) i.e.

(SD)

$$\min_{u \geqslant 0} \left\{ \begin{array}{lll} \max & cx & \\ \text{s.t.} & uAx \geqslant ub & \\ & x_j \in \{0,1\} & j \in J = \{1, \ldots, n\} \end{array} \right\}$$

To generate this multiplier, we use a quasi-subgradient algorithm for $m > 2$ [13], or a dichotomic method for $m = 2$, described in more detail in [20,40].

More simply, structural multipliers that depend only on the 0-1 MKP's data can be employed:

$$u_i := \begin{cases} \left(\sum_{j=1}^{n} A_i^j - b_i\right) \Big/ \sum_{j=1}^{n} A_i^j & \text{if } \sum_{j=1}^{n} A_i^j \geqslant b_i, \\ 0 & \text{otherwise.} \end{cases}$$

6.2. Oscillation parameters

In this subsection we detail how the oscillation parameters are controlled. The depth of oscillations on both sides of the boundary can be managed in several ways [32]. In [34] the depth of the oscillations is predetermined, periodic and symmetrical on both sides of the boundary of the feasible domain. Fixed parameters control the oscillations; a span parameter counting the ADD/DROP moves is varied between one and an upper bound; two other parameters $p1$ and $p2$ define the periodicity. Fig. 2 gives an idea of the shape of oscillations in the case of FHP5 instance [19], $m = 10$, $n = 20$, $p1 = 3$, $p2 = 7$.

Our choice is more flexible and differs on both sides of the boundary of the feasible domain. The oscillation appears at two levels, in the intensification phase and in the diversification phase.

In the intensification phase, the oscillation is controlled by the parameter V of the COMPLEMENT procedure. Once the critical level is reached from the feasible side (which corresponds to the step C2), an intensive search is adopted around it in the promising zone. This is achieved by calling COMPLEMENT procedure with the parameter $V = J1(x)$. In this case, the search is focused on the feasible neighborhood of the current

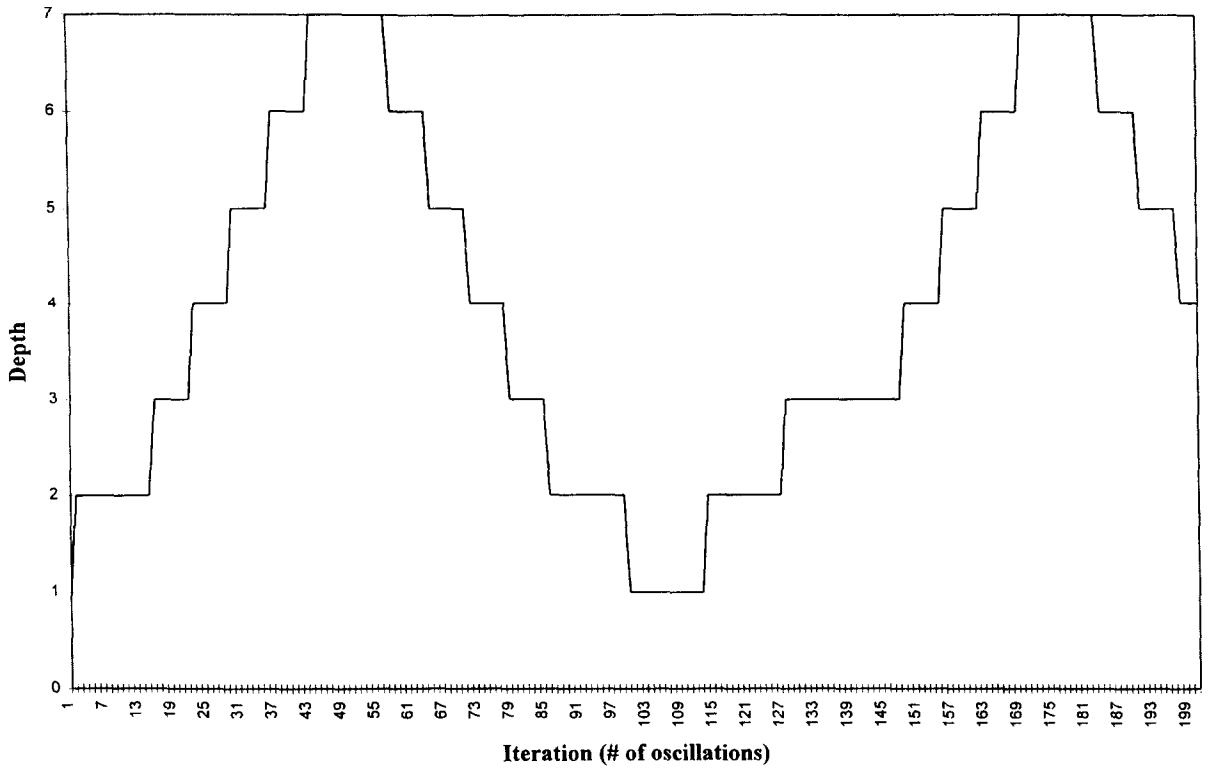

Fig. 2. Depth of oscillations with GK method.

critical level x. Symmetrically, when the promising zone is reached from the infeasible side (which corresponds to the step D2), the COMPLEMENT procedure is called with the parameter $V = J0(x)$. In this case, the search explores intensively an infeasible neighborhood of the current critical level x.

In the diversification phase, the depth of the oscillation is controlled by parameters *depth* and *near_feasible* of TS_DROP and TS_INFEASIBLE_ADD procedures, respectively. After each intensification phase that generates a feasible solution at the boundary, the diversification phase is launched with either the TS_INFEASIBLE_ADD procedure or the TS_DROP procedure. These procedures allow the search to create more pronounced departures from the critical level.

On the *feasible side*, the amplitude of oscillations is given by the value of the parameter *depth* (in practice this value is fixed at 1). However, it can happen that after a D3 step, the current solution remains unchanged during the rest of the iteration (step Ci: $i = 1, 2, 3$; D1 and D2). Consequently, more than two D3 steps can be successively performed, and in this case the depth of amplitude gives beyond the predetermined value of depth.

On the *infeasible side*, a solution generated at each iteration is allowed to become infeasible with respect to some of the knapsack constraints. In our implementation three types of knapsack constraints are considered.

- *Surrogate constraint (TS1)*: the infeasibility of the generated solutions is controlled within a surrogate constraint $uAx \leqslant ub$ associated with a multiplier u (see Section 6.1); then x *is near feasible* becomes $uAx \leqslant ub$.
- *Violated constraints permutation (TS2)*: the solutions must satisfy at least one constraint $A_s x \leqslant b_s$ 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* (i.e., that *near_feasible* = true) becomes $A_s x \leqslant b_s$ with $s := k$ modulo m.

- *Least saturated constraint (TS3)*: the solutions generated by this variant satisfy also at least one constraint $A_s x \leqslant b_s$, where s is controlled by the state of the search. Recall the before calling TS_INFEASIBLE_ADD, at the iteration k, the current solution denoted x^k is assumed to be on the feasible boundary (critical level). The constraint to satisfy, denoted by s, is determined as the least saturated constraint of the feasible solution x^k. Then in this case "x is near_feasible", means $A_s x \leqslant b_s$ with $s :=$ argmax $\{b_i - A_i x^k: i = 1, \ldots, m\}$.

Figs. 3 and 4 compare the depth of oscillations of a feasible TS version, called TS0 (in this case the depth of oscillations on the infeasible side is equal to 0), with the three infeasible TS versions described above TSi, $i = 1, 2, 3$. The curves correspond to the FHP5 instance. (TL length is equal to 7 and a structural multiplier is used.) Increasing (resp. decreasing) parts of the curves correspond to ADD (resp. DROP) moves, otherwise to the number of variables fixed to 1.

In the TS0 and TS1 versions, the oscillations have a more regular behavior than in the TS2 and TS3 variants. The boundary on the infeasible side of TS0 and TS1 is predetermined and fixed during the search. In fact, the oscillations are embedded in the feasible domain of the 0-1 MKP instance (TS0) or of its surrogate relaxation associated with the multiplier u (TS1): On the contrary, in TS2 and TS3 the boundary is determined by the state of the current solution.

6.3. Memory management

The tabu status of a potential move is determined through the integration of recency-based and frequency-based memory information. In the 0-1 MKP, the attribute of an elementary DROP/ADD move is identified by the index j of the binary variable x_j. The recency information are recorded in a tabu circular list denoted TL, which is updated at the extremities of each oscillation execution. In our implementation, we use a deterministic TL management (see Section 7). The frequency memory information is

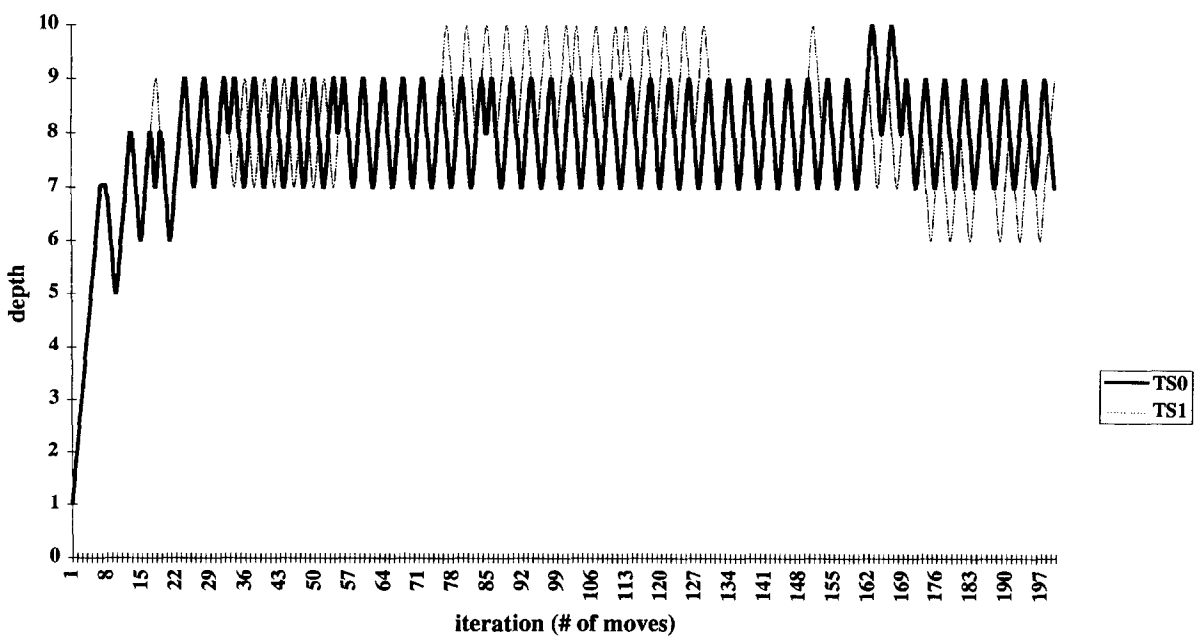

Fig. 3. Depth of oscillations for TS0 and TS1.

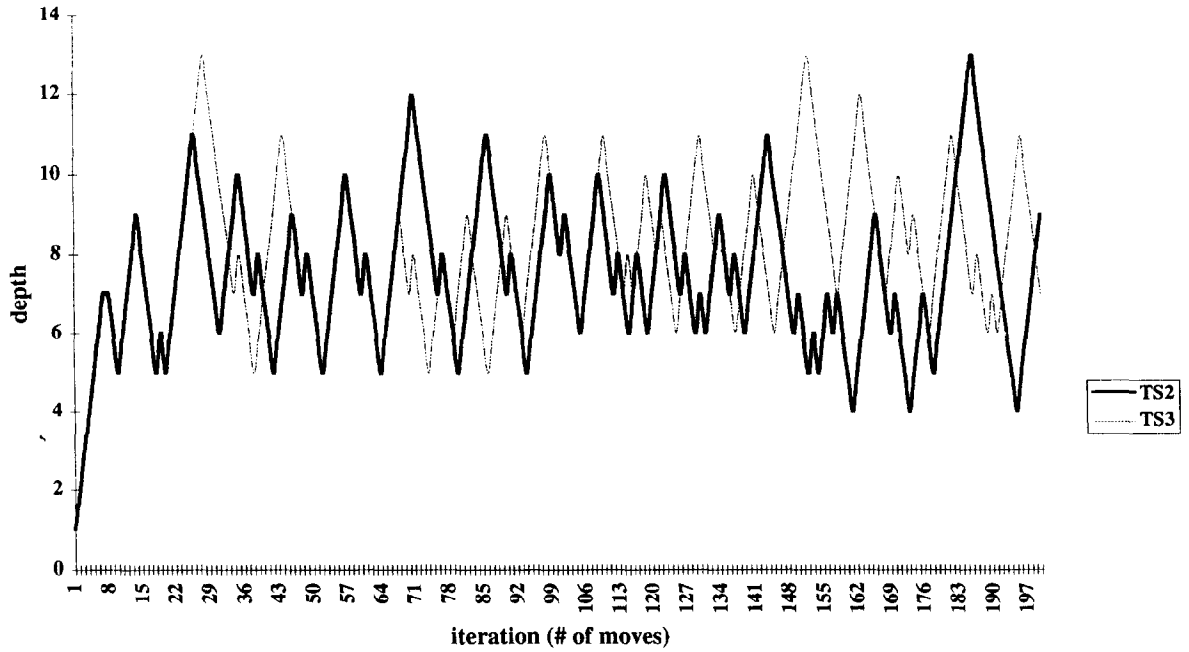

Fig. 4. Depth of oscillations for TS2 and TS3.

contained in two vectors (H^+ and H^*) of size n. These vectors keep track of all the transitions performed during the moves: $H^+(j)$ (respectively $H^-(j)$) is incremented by 1 if the variable x_j is added (resp. dropped). The frequency-based memory information is used to reinforce the intensification and diversification phases. Alternatively, intensification and diversification are launched as soon as no improvement occurs during a fixed number of iterations (in our experiment this number is set to 21). The intensification strategy performs a local search starting from the best feasible solution. The solution space is then reduced by fixing the variables associated with the highest values in H^+ and H^-. By contrast, the diversification strategy tries to drive the search into new unexplored regions by fixing variables of the current solution associated with the lowest values in H^+ and H^-. These steps of intensification and diversification are specified by calling the COMPLEMENT procedure with the parameter V set to J and the TL parameter set to $\emptyset$ (which does not make recource to memory).

7. Numerical experiments

In this section we present computational results obtained with different variants of *TS_Oscillation* Algorithm. All the procedures have been coded using Borland's C++ 7.0. The runs are performed on a compatible personal computer with a 80486 DX2-80 central processing unit. All running times are given in seconds including the display of the state of search at each iteration. One iteration corresponds to one oscillation and the maximum number of iterations is at most $10n$ in the *TS_Oscillation* Algorithm.

Before giving our results, some remarks about the test problems are in order. We tested our approach on two sets of instances of 0-1 MKP. The first one is issued from Fréville and Plateau [17,19] and is composed of 54 instances. The second set is due to Glover and Kochenberger [34] and is composed of 24 instances.

As in [34,37,41], optimal solutions are successfully obtained for each of the instances of the first set. Optimal solutions are known for all the instances of the second set, except for the last seven

problems when the size n varies from 100 to 500. So, the results reported in Table 2 only concern these seven problems which are particularly difficult to solve for Branch and Bound algorithms like LINDO. Glover and Kochenberger reported in [34] that in some cases LINDO ran for more than four consecutive days on PC 486 without terminating with a proven optimal solution. For six (except GK 18) of these seven problems the GK approach is superior to the Branch and Bound algorithm both in solution quality and computational time (a speed difference on average of about 50–1). The best solution they obtained for each problem is given in the column quoted GK, while the column quoted $v(S)$ indicates an upper bound of the optimal value $v(P)$. This upper bound is an approximation of the SD provided by a quasi-subgradient algorithm [13].

The results of Glover and Kochenberger [34] are compared with the different versions of TS proposed in this paper. For each version TSi ($i = 0, \ldots, 3$), we give the minimum and the maximum values obtained over 12 runs of each problem. These runs are parametrized by the size of TL varying from 19 to 41 by step of size 2. We compare also the influence of the multipliers u^0 and u^1, where u^0 corresponds to a near optimal multiplier of the surrogate dual and u^1 to a structural multiplier (see Section 6.1). This table confirms that the optimal multiplier of the SD gives better results than the structural one. Globally, our approach generates solutions of high quality, the best of them being significantly superior to those given in [34].

Moreover, best results have been obtained by increasing the TL size |TL|. For the problem GK 18, the value 4525 is generated with $|TL| = 60$ by TS1 and by TS2 with $|TL| = 80$. For the problem GK24, the value 9065 is reached by TSi ($i = 1, 2, 3$) with $|TL| \in \{100, 200\}$. These first numerical experiments show the fruitfulness of an oscillation strategy that exploits adequate dual information for the class of problems considered.

To provide additional information on the time required by our TS_Oscillation Algorithm, we give the average time of one iteration for each of the four versions (TSi: $i = 0, \ldots, 3$). An iteration corresponds to an oscillation defined by the steps Ci ($i = 1, 2, 3$) and the steps Di ($i = 1, 2, 3$). Then the maximum number of iterations, which is fixed at $10n$, determines the total CPU time consumed by the method. The 78 test problems considered in our study have been divided into subsets according to a size parameter $n \times m$.

For each subset, several runs of our TS_Oscillation Algorithm have been performed with different values of the parameters; the length of the TL varies from $\lfloor n/5 \rfloor$ to $(n - \lfloor n/5 \rfloor)$ by a step size equal to $\lfloor n/5 \rfloor$ and the multiplier in the COMPLEMENT procedure is either an approximation of

Table 2
Seven diffcult problems

Problem	m	n	$v(S)$	GK	u	TS0		TS1		TS2		TS3	
						Min	Max	Min	Max	Min	Max	Min	Max
GK18	25	100	4546	4524	u^0	4516	4523	4516	4519	4517	4524	4516	4523
					u^1	4510	4523	4512	4520	4506	4516	4501	4514
GK19	25	100	3887	3589	u^0	3857	**3866**	3857	3862	3856	3866	3858	3862
					u^1	3850	3856	3849	3861	3843	3856	3840	3859
GK20	25	100	5198	5177	u^0	5173	**5177**	**5177**	**5177**	5176	**5177**	5173	**5177**
					u^1	5161	5172	5166	5171	5163	5174	5162	5171
GK21	25	100	3220	3194	u^0	3194	**3195**	3194	**3195**	3194	3194	3194	**3195**
					u^1	3183	3193	3182	3190	3181	3186	3174	3194
GK22	25	100	2544	2515	u^0	2517	2517	2517	**2521**	2517	**2521**	2517	**2521**
					u^1	2506	2516	2505	2516	2507	2515	2505	2514
GK23	15	200	9245	9231	u^0	9227	9230	9227	9229	9228	**9231**	9227	9228
					u^1	9215	9223	9215	9223	9211	9229	9206	9221
GK24	25	500	9080	9057	u^0	9057	**9062**	9057	9061	9053	9057	9053	9056
					u^1	8937	8966	8946	8969	8809	8915	8827	8889

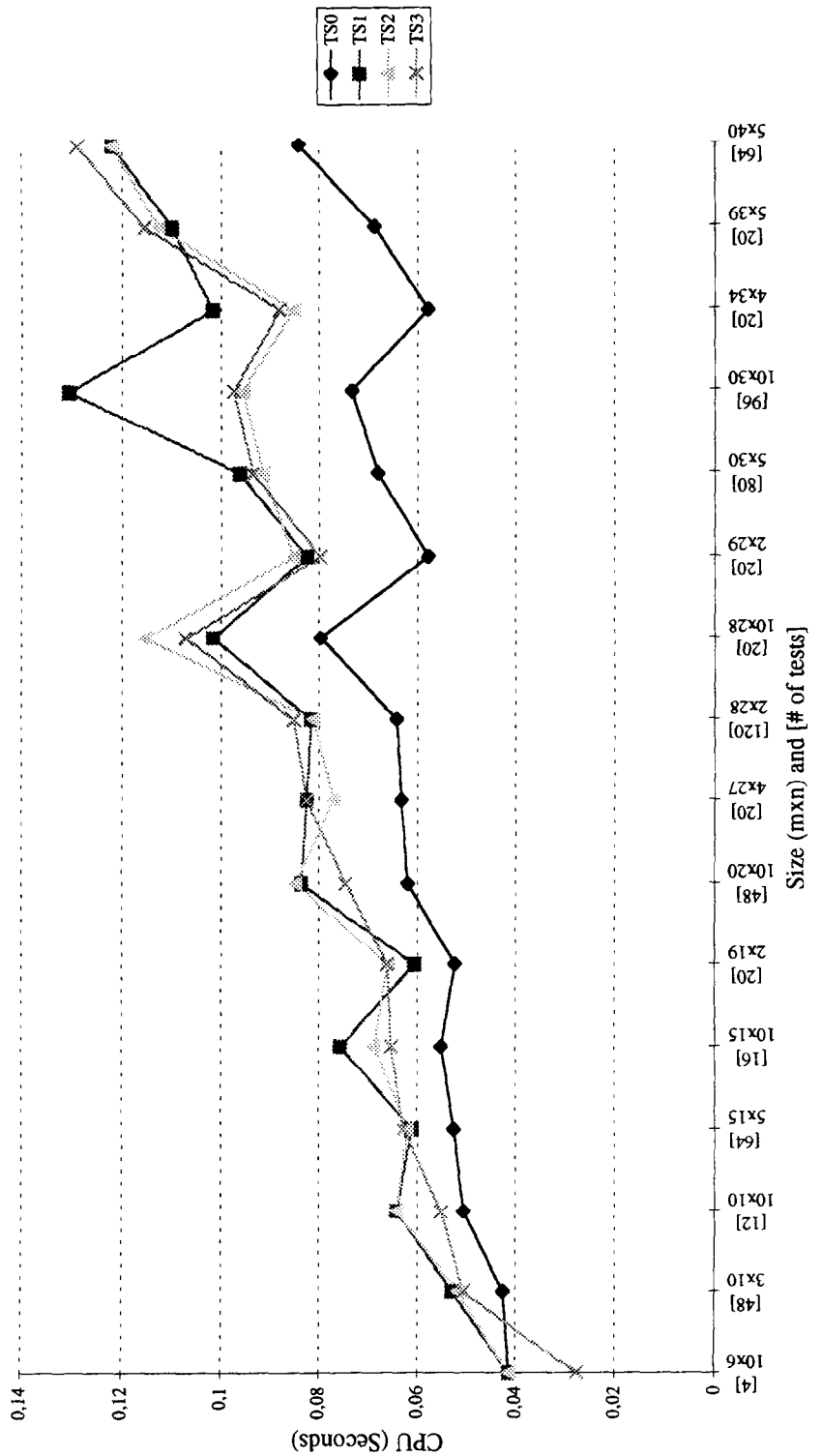

Fig. 5. CPU time on average by iteration.

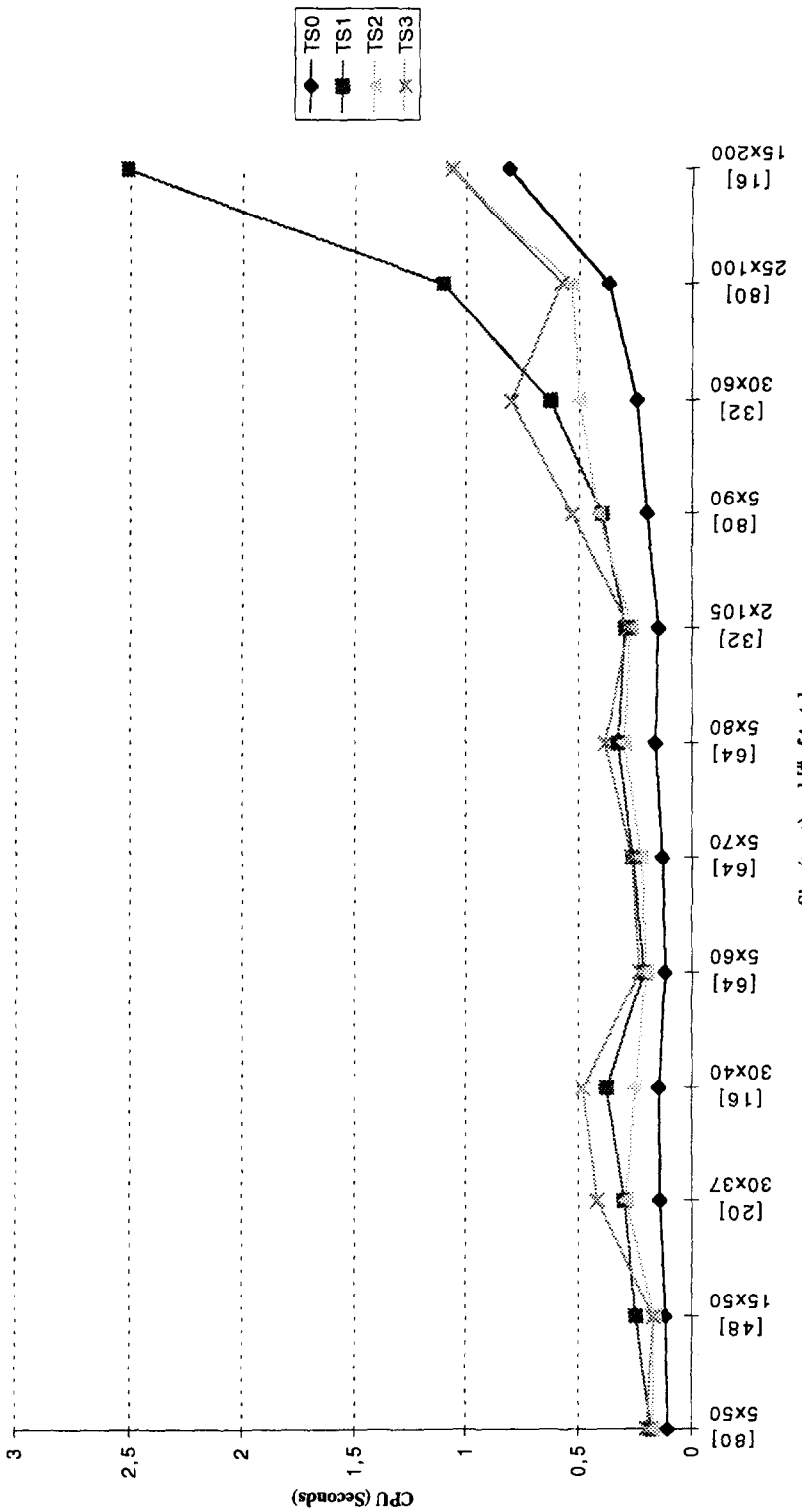

Fig. 6. CPU time on average by iteration.

Table 3
CPU time on average by iteration for GK24

Problem	$m \times n$	TS0	TS1	TS2	TS3
GK24	25×500	3.77	4.11	6.49	6.86

the optimum of the SD or a structural evaluation. The results are reported in Figs. 5 and 6 with the number of runs quoted into brackets for each subset $m \times n$.

To render the curves more legible, the largest instance GK24 has not been plotted in Fig. 6. The following Table 3 explicitly gives its time on average by iteration.

It is not surprising that the feasible version TS0 is the least time consuming, because the two steps C3 and D1 are never performed during an iteration. At the opposite extreme the most time-consuming version is TS1 where infeasibility is controlled with a surrogate constraint.

Improvements can be made in our different procedures to decrease the time of execution significantly. For example, if the multiplier u is not dependent on the state of the current solution x, the ratio c_j/uA^j can be sorted in increasing (decreasing) order for adding (dropping) a variable, before the loop statement.

The following Table 4 illustrates the percentage of success and the average relative deviation from optimal solution obtained for each version of TS_Algorithm. Problems considered here are only those for which optimal solutions are known. The number of runs for each problem is indicated in Figs. 5 and 6.

The infeasible versions of TS_Algorithm are better than the feasible variant TS0 with respect to the quality of the solutions obtained. Moreover, the flexibility of the TS2 and TS3 versions, induced by the link between the depth of the oscillations and the current state of the search, provides a more relevant strategy than in the TS1 case.

In the COMPLEMENT procedure, the use of dual information (particularly dual surrogate information) is more promising than the use of structural multipliers. Indeed, we observed that the surrogate multipliers that approximate the SD are best in both solution quality and execution

Table 4
Percentage of success and deviation

	Percentage of success (%)	Deviation (%)
TS0	57.5	1
TS1	68.1	0.5
TS2	83.8	0.2
TS3	80.9	0.2

time. Almost 80% of the optimal solutions of 71 problems considered, with different sizes of TL, are obtained in only $2n$ iterations.

The SD must be solved to get these multipliers. This induces an additional computational cost, since algorithms used to solve the dual surrogate need the solution of a knapsack problem at each iteration. Our results indicate that the extra time and effort is worthwhile.

8. Conclusion

In this paper we report the implementation of an efficient TS method for solving the 0-1 MKP. Our method has been tested on standard test problems from the literature. Optimal solutions are successfully obtained for each of 74 instances and the previously best known solutions are improved for five of the last seven instances. Our numerical results confirm the merit of tabu tunneling approaches to generate solutions of high quality for 0-1 multiknapsack problems. Moreover, our results (like those of Glover and Kochenberger) establish that the oscillation strategy is efficient to balance the interaction between intensification and diversification strategies of TS.

References

[1] R. Aboudi, K. Jörnsten, Tabu search for general zero-one integer programs using the pivot and complement heuristic, ORSA Journal of Computing 6 (1994) 82–93.
[2] E. Balas, C.H. Martin, Pivot and complement – A heuristic for 0-1 programming, Management Science Research 26 (1980) 86–96.
[3] R. Battiti, Reactive search: towards self-tuning heuristics, Congress of Applied Decision Technologies, Brunel, London, UK, April 1995.
[4] R. Battiti, G. Tecchiolli, The reactive tabu search, ORSA Journal on Computing 6 (1994) 126–140.

[5] R. Battiti, G. Tecchiolli, Local search with memory: benchmarking RTS OR Spectrum 17 (1995) 67–86.
[6] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.
[7] E.D. Chajakis, M. Guignard, A model for delivery of groceries in vehicles with multiple compartments and Lagrangean approximation schemes, Congreso Latino Ibero-Americano de Investigation de Operaciones e Ingenieria de Sistemas, Mexico City, October 1992.
[8] I. Charon, O. Hudry, The noising method: A new method for combinatorial optimization, Operations Research Letters (1993) 133–137.
[9] F. Dammeyer, S. Voß, Dynamic tabu list management using the reverse elimination method, Annals of Operations Research 41 (1993) 31–46.
[10] A. Drexl, A simulated annealing approach to the multiconstraint zero-one knapsack problem, Computing 40 (1988) 1–8.
[11] G. Dueck, New optimization heuristics - The great deluge algorithm and the record-to-record travel, IBM Technical report TR 89.06/011, 1989.
[12] G. Dueck, T. Scheurer, Threshold accepting: a general purpose optimization algorithm, Journal of Computational Physics 90 (1990) 161–175.
[13] H.E. Dyer, Calculating surrogate constraints, Mathematical Programming 12 (1980) 255–278.
[14] D. Fayard, G. Plateau, An exact algorithm for the 0-1 collapsing knapsack problem, Discrete Applied Mathematics 49 (1994) 175–187.
[15] T.A. Feo, M.G.C. Resende, S.H. Smith, A greedy randomized adaptive search for maximum independent set, Operations Research 42 (1994) 860.
[16] A. Fréville, Contribution à l'Optimisation en Nombres Entiers, Habilitation à Diriger des Recherches, Université de Paris XIII, France, 1991.
[17] A. Fréville, G. Plateau, Méthodes Heuristiques Performantes pour les Programmes en Variables 0-1, Working paper, ANO-91, Université des Sciences et Techniques de Lille, France, 1982.
[18] A. Fréville, G. Plateau, Heuristics and reduction methods for multiple constraints 0-1 linear programming problems, European Journal of Operational Research 24 (1986) 206–215.
[19] A. Fréville, G. Plateau, Hard 0-1 multiknapsack test problems for size reduction methods, Investigacion Operativa 1 (1990) 251–270.
[20] A. Fréville, G. Plateau, An exact surrogate dual search for the 0-1 bidimensional knapsack problem, European Journal of Operational Research 68 (1993) 413–421.
[21] A. Fréville, G. Plateau, An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem, Discrete Applied Mathematics 49 (1994) 189–212.
[22] A. Fréville, G. Plateau, The 0-1 bidimensional knapsack problem: towards an efficient high level primitive tool. Journal of Heuristics 2 (1996) 147–167.
[23] B. Gavish, H. Pirkul, Allocation of databases and processors in a distributed data processing, in: J. Akola (Ed.), Management of Distributed Data Processing, North-Holland, Amsterdam, 1982, pp. 215–231.
[24] B. Gavish, H. Pirkul, Models for computer and file allocation in distributed computer networks, Working paper, The Graduate School of Management, the University of Rochester, 1983.
[25] B. Gavish, H. Pirkul, Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality, Mathematical Programming 31 (1985) 78–105.
[26] C.E. Gearing, W.W. Swart, T. Var, Determining the optimal investment policy for the tourism sector of a developing country, Management Science 20 (1973) 487–497.
[27] P.C. Gilmore, R.E. Gomory, The theory and computation of knapsack functions, Operations Research 14 (1966) 1045–1074.
[28] F. Glover, A multiphase dual algorithm for the 0-1 integer programming problem, Operations Research 13 (1965) 879–919.
[29] F. Glover, Heuristics for integer programming using surrogate constraints, Decision Sciences 8 (1977) 156–166.
[30] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 19 (1986) 533–549.
[31] F. Glover, Tabu search, Part 1, ORSA Journal on Computing 1 (1989) 190–206.
[32] F. Glover, Tabu thresholding: improved search by nonmonotonic trajectories, ORSA Journal On Computing 7 (1995) 426–442.
[33] F. Glover, Tabu search fundamentals and uses, Working paper, University of Colorado, USA, 1995.
[34] F. Glover, G.A. Kochenberger, Critical events tabu search for multidimensional knapsack problems, in: I.H. Osman, J.P. Kelly (Eds.), Metaheuristics: The Theory and Applications, Kluwer Academic Publishers, Dordrecht, 1995, pp. 407–428.
[35] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Dordrecht, 1997.
[36] F. Glover, A. Lokkentangen, Solving zero-one mixed integer programming problems using tabu search, Working paper, University of Colorado, USA, 1994; updated version, by Lokketangen and Glover (to appear in this issue).
[37] F. Glover, A. Lokketangen, Tabu search for zero-one mixed integer programming with advanced level strategies and learning, Working paper, University of Colorado Boulder, USA, 1995 (forthcoming in International Journal of Operations and Quantitative Management).
[38] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA 1989.
[39] S. Grolimund, J.G. Ganascia, Integrating case-based reasoning and tabu search for solving optimization prob-

lems, Proceedings of the International Conference on Case-Base Reasoning, Lectures Notes in Computer Science, Springer, Berlin, 1995.

[40] S. Hanafi, Contribution à la Résolution de Poblèmes Duaux de Grande Taille, Thèse de Doctorat, Université de Valenciennes, France, 1993.

[41] S. Hanafi, A. Fréville, A. El Abdellaoui, Comparison of heuristics for the 0-1 multidimensional knapsack problem, in: I.H. Osman, J.P. Kelly (Eds.), Metaheuristics : Theory and Applications, Kluwer Academic Publishers, Dordrecht, 1995, pp. 449–466.

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

[43] J.H. Holland, Adaptation in Natural and Artificial Systems, An Introduction Analysis with Applications to Biology Control and Artificial Intelligence, University of Michigan Press, Ann Arbor, 1975.

[44] J.J. Hopfield, D.W. Tank, Neutral computation of decisions in optimization problems, Biological Cybernetics 52 (1985) 141–152.

[45] S. Kirkpatrick, C.D. Gelatt, M.P. Vechi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[46] J.S. Lee, M. Guignard, An approximate algorithm for multidimensional zero-one knapsack problems – A parametric approach, Management Sciences 34 (1988) 402–410.

[47] A. Lokketangen, F. Glover, Probabilistic move selection in tabu search for zero-one mixed integer programming problems, in: I.H. Osman, J.P. Kelly (Eds.), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, Dordrecht, 1995, pp. 467–488.

[48] A. Lokketangen, K. Jörnsten, S. Storoy, Tabu search within a pivot and complement framework, International Transactions of Operations Research 1 (1994) 305–316.

[49] J. Lorie, L.J. Savage, Three problems in capital rationing, Journal of Business 28 (1955) 229–239.

[50] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21 (1953) 1087–1091.

[51] M. Ohlssen, C. Peterson, B. Soderberg, Neural networks for optimization problems with inequality constraints: the knapsack problem, Neural Computation 5 (1993) 331–339.

[52] I.H. Osman, J.P. Kelly, Meta-heuristics: an overview, in: I.H. Osman, J.P. Kelly (Eds.), Metaheuristics: Theory and Applications, Kluwer Academic Publisher, Dordrecht, 1995, pp. 1–22.

[53] C.C. Petersen, Computational experience with variants of the Balas algorithm applied to the selection of R&D projects, Management Science 13/9 (1967) 736–750.

[54] S. Senju, Y. Toyoda, An approach to linear programming with 0-1 variables, Management Sciences 15B (1968) 196–207.

[55] W. Shih, A branch and bound method for the multiconstraint zero-one knapsack problems, Journal of Operations Research Society 30 (1979) 369–378.

[56] A. Straszak, M. Libura, J. Sikorski, D. Wagner, Computer-assisted constrained approval voting, Group Decision and Negociation 2 (1993) 375–385.

[57] J. Thiel, S. Voβ, Solving multiconstraint zero-one knapsack problems with genetic algorithms, Working paper, TH Darmstadt, Germany, September 1992.

[58] M. Toulouse, T.G. Crainic, M. Gendreau, Communication issues in designing cooperative multi-thread parallel searches, Working paper, Centre de recherche sur less Transports, Université de Montréal, 1995.