CRITICAL EVENT TABU SEARCH FOR MULTIDIMENSIONAL KNAPSACK PROBLEMS

Fred Glover
Graduate School of Business, Box 419
University of Colorado at Boulder
Boulder, Colorado, 80309-0419
E-Mail: fred.glover@colorado.edu

Gary A. Kochenberger
College of Business
University of Colorado at Denver
Denver, Colorado 80217-3364
E-Mail: gkochenberge@castle.cudenver.edu

Abstract

We report a new approach to creating a tabu search method whose underlying memory mechanisms are organized around "critical events." A balance between intensification and diversification is accomplished by a strategic oscillation process that navigates both sides of the feasibility boundary, and serves to define the critical events. Surrogate constraint analysis is applied to derive choice rules for the method. Computational tests show the approach performs more effectively than previous heuristics for multidimensional knapsack problems, obtaining optimal solutions for all problems in a standard testbed.

I. H. Osman et al. (eds.), *Meta-Heuristics*
© Kluwer Academic Publishers 1996

Introduction.

Considerable progress has been made over the past decade in developing and testing search techniques for combinatorial optimization problems. Various forms of genetic algorithms, simulated annealing, and tabu search have all been reported to perform well in diverse problems settings. Tabu search (TS) has proved highly effective in various implementations across a wide range of applications, and forms the foundation of the approach of this paper. Background on applications of TS and elements responsible for its successes can be found in recent survey papers [8, 10, 11].

The distinguishing characteristic of tabu search is the use of adaptive memory structures which are accompanied by associated strategies for exploiting information during the search. The adaptive memory component of TS typically incorporates recency-based and frequency-based memory defined over varying short term and long term spans of the search history. This memory operates through control mechanisms of tabu restrictions and aspiration criteria, and associated penalties and inducements to modify move evaluations. By such adaptive designs, which abstract certain processes conjectured to operate in human problem solving, this memory is sometimes contrasted with the "rigid" memory structures of branch and bound, and the "memoryless" designs of a variety of other approaches.

The choice rules of tabu search are highly aggressive, seeking "best moves" at each step (from those allowed by its restrictions and made available from its history). To apply this aggressive orientation judiciously, the approach characteristically relies on strategies to isolate subsets of desirable moves (rather than to exhaustively examine all alternatives at each iteration), and utilizes choice criteria that make use of a notion of *influence* (which includes elements of criticality and patterning), in addition to drawing on the outcomes of search history. A principal theme is to create an effective interplay between intensification and diversification, that is, between approaches designed to identify and reinforce attributes of good solutions and approaches designed to draw the search into promising new regions. A general treatment of tabu search and its principal strategies may be found, for example, in [8] and [9].

In this paper we focus on a special subset of these elements and report on a new method for creating an effective search process based on a flexible memory structure that is updated at *critical events*. The tabu status of a potential move is determined through the integration of recency-based and frequency-based memory information. A balance between intensification and diversification is accomplished by a strategic oscillation scheme that probes systematically to varied depths on each side of the feasibility boundary. These oscillations, coupled with dynamic tabu information, guide the search process toward different critical events.

The approach illustrated here applies to many types of problems. However, we specifically will be concerned with an implementation designed to solve multidimensional knapsack (MK) problems, which take the form:

$$\begin{array}{ll} \textbf{MK:} & \text{maximize } cx \\ \text{subject to} & \\ & Ax \leq b \\ & x \text{ binary} \end{array}$$

The matrix A and the vectors b and c consist of real-valued constants that satisfy

$A \geq 0$ and $b, c \geq 0$.

1. Basic Notions.

Our approach is representative of a class of strategic oscillation methods that proceed by alternating between constructive and destructive phases. In the present setting, a constructive phase corresponds to one that progressively sets variables equal to 1, while a destructive phase corresponds to one that progressively sets variables to 0.

Within this framework, we organize a strategic oscillation as follows. We first sketch the ideas in overview, and then diagram the steps of the approach. We introduce a parameter *span* that indicates the amplitude (or depth) of the oscillation about the feasibility boundary, measured in the number of variables *added* (set to 1) when proceeding from the boundary into the infeasible region and in the numbers of variables *dropped* (set to 0) when proceeding from the boundary into the feasible region. Although the depth in these two directions need not be the same (and in fact asymmetric or even "one-sided" oscillations can be best for certain problems) we focus here on treating them as equal.

We begin with span equal to 1, and gradually increase it to a limiting value. For each value of span, a series of alternating constructive and destructive phases is executed before progressing to the next value. At the limiting point, we begin to gradually decrease span, allowing again a series of constructive and destructive phases for each span value before progressing to the next. When span reaches a value of 1, we begin once more a gradual increase in span. The manipulation of the span parameter may be viewed as an *outer oscillation* that contains the oscillation of constructive and destructive moves (for a given value of span) within it.

1.1 Method in Detail.

Let x* denote the best solution found so far; that is, the feasible solution that gives a maximum value for cx. Until x* has been determined, cx* is taken to be a large negative number. We assume that setting all $x_j = 1$ is infeasible, else the problem is trivially solved. The terms *tabu* and *non_tabu* will be given operational definitions subsequently. Our strategic oscillation method, then, consists of the following:

Initialization

Step 0: Begin with $x = 0$ or $x = e$ and set count_span = 0. If $x = 0$, set feasible = .true. and go to the Constructive Phase. If $x = e$, set feasible = .false. and go to the Destructive Phase.

Constructive Phase: (move from feasible to infeasible)

Step C1: (feasible = .true.)

(1) If no component x_j of x can be increased from 0 to 1 except by violating feasibility, then:

(a) if $cx > cx^*$, let $x^* = x$

(b) set feasible = .false. and go to step C2.

(2) If condition (1) does not hold, then choose an x_j to increase from 0 to 1, such that the move maintains feasibility, and return to the start of step C1.

Step C2: (feasible = .false.)

Set count_span = count_span + 1

(1) If count_span > span, or if there are no x_j available to change from 0 to 1, go to the Transfer Phase.

(2) If condition (1) does not hold, choose an x_j to increase from 0 to 1 and return to the start of Step C2.

Transfer Phase:

Step T1: Set count_span = 0. Change the value of span if appropriate (by a rule to be identified later).

Step T2: Go to the Destructive Phase if the last phase was a Constructive Phase. Else, go to the Constructive Phase.

```
Destructive Phase: (move from infeasible to feasible)
    Step D1: (feasible = .false.)
        (1)  Select an $x_j$ to change from 1 to 0.
        (2)  If the solution produced by (1) is infeasible,
             return to the start of Step D1. Otherwise, set
             feasible = .true. and:
             (a) if cx > cx*, let x* = x.
             (b) go to Step D2.

    Step D2: (feasible =.true.)
        Set count_span = count_span+1
        (1)  if count_span > span, or if there are no $x_j$
             available to change from 1 to 0, go to the
             Transfer Phase.
        (2)  If condition (1) does not hold, choose an $x_j$ to
             decrease from 1 to 0 and return to the start of
             Step D2.
```

The preceding method terminates whenever a selected number of iterations have been performed without finding an improved x*, or simply after a total iteration limit (number of Transfer Phase executions) has been reached.

1.2 Surrogate Constraint Choice Rules.

As in a variety of strategic oscillation approaches, we make use of surrogate constraint information to guide the decisions of the method. In particular, the choice of a variable to add (steps C1(2) and C2(2)) and of a variable to drop (steps D1(1) and D2(2)) is determined by a standard surrogate constraint evaluation. (For background on surrogate constraints and their uses in choice rules, see Glover [6, 7].) We dynamically form surrogate constraints, based upon the current solution, as normalized (nonnegative) linear combinations of the problem constraints. At each iteration, we compute ratios of the objective function coefficients to their associated surrogate constraint coefficients. During a Constructive Phase, we add variables with maximum ratios and during a Destructive Phase we drop variables with minimum ratios. When tabu restrictions are enforced, this ratio information is augmented by recency and frequency information.

The surrogate constraints used to determine the choice rules are formed by first normalizing (or otherwise weighting) and then summing various constraints. We currently employ three different surrogates depending on the feasibility status of the current solution vector and the phase of the search. In all cases we compute

$$b_i' = b_i - \Sigma(a_{i,j} : \text{for } j \text{ with } x_j = 1).$$

As long as the current solution is feasible, the weight w_i for constraint i is chosen to be $1/b_i'$. When the search process enters the infeasible region, we choose weights by one of two methods:

a. If $b_i' > 0$, set $w_i = 1/b_i'$
   If $b_i' \leq 0$, set $w_i = 2 + |b_i'|$

b. If $b_i' \geq 0$, set $w_i = 0$
   If $b_i' < 0$, set $w_i = 1/(|b_i'| + \Sigma(a_{i,j}: \text{for } j \text{ with } x_j = 0))$

We have experimented with both methods and currently use method (a) in the Constructive Phase and method (b) in the Destructive Phase. The rationale for the above procedures is to exaggerate the influence of the most violated constraints and thus encourage searches "near" critical solutions. (Our testing to date has not conclusively demonstrated whether one method is preferred to the other.)

Let $\Sigma\, s_j x_j \leq s_o$ denote the resulting surrogate constraint, i.e., $s_j = \Sigma w_i a_{ij}$ and $s_o = \Sigma w_i b_i$. The choice rule for the constructive phase then selects the variable x_j to change from 0 to 1 in order to

$$\text{Maximize } (c_j/s_j\colon\ x_j = 0)$$

while the choice rule for the destructive phase selects the variable x_j to change from 1 to 0 in order to

$$\text{Minimize } (c_j/s_j\colon\ x_j = 1).$$

These rules are modified by the TS restrictions and penalties subsequently indicated. In the next sections, we describe how we accomplish this in our implementation.

2. Tabu Memory.

The essence of the method rests upon the scheme for defining tabu status, and hence that guides the oscillations productively. In our implementation, the tabu status of a potential move (adding or dropping a variable) is determined by recency and frequency information gathered as the search process encounters critical events. For MK problems, we define a critical event to be the construction of a complete (feasible) solution by the search process at the feasibility boundary. That is, critical events correspond to solutions obtained by the Constructive Phase at the final moment before going infeasible, and by solutions obtained by the Destructive Phase at the first moment of regaining feasibility.

At critical events, we also include steps for generating additional trial solutions which are further candidates for x^*. In the Constructive Phase, we proceed as follows. As variables are chosen to be changed from 0 to 1, a move is finally made that drives the search infeasible. When such a step is imminent, a trial solution can be identified if some variable can be found to add (change from 0 to 1) that "fits" within the constraints even though it contributes less to the objective function than the one normally chosen (that produces infeasibility). Thus, at such a juncture, the variables are examined, in order of decreasing objective function coefficients, for the first such variable that can be added while maintaining feasibility. The trial solution choice does not replace the customary choice (in the standard variant we employ), but simply provides a solution that is recorded if it improves the best one currently known.

The second trial solution (at a critical solution in the Constructive Phase) is generated by retaining the regularly selected move that produced infeasibility (as noted) and searching for a variable to drop (other than the one just selected to add) that will re-establish feasibility. Candidate variables to drop for the purpose of generating this second trial solution are examined in order of increasing objective function coefficients.

Corresponding to the two trial solutions generated at the critical level of the Constructive Phase, we similarly generate two such solutions at the critical level of the Destructive Phase. The rules are the mirror images of the above steps. Note that trial solutions generated in this way represent the use of an aspiration criterion since the solutions are found by temporarily ignoring tabu information.

2.1 Using Recency and Frequency Information.

In order to influence the search by recency information, we record (in a circular list) the last t solutions obtained at critical events, where in our implementation t is a simple function of problem size that takes values in the range of 3 - 12. Using this record of the last t critical solutions, which range from x(last) to x(last-(t-1)), we maintain a short term recency tabu vector (TABU_R) that is the sum of the last t solutions. That is, each time a new x(last) is identified, we set

$$\text{TABU\_R} = \text{TABU\_R} + x(\text{last}) - x(\text{last-t}).$$

In a like manner, long term frequency information is captured for use in the search process by maintaining another vector (TABU_F) which is the sum of all critical solutions encountered to date (rather than the last t critical solutions).

As previously noted, during a Constructive Phase, variables are added until the span parameter indicates it is time to switch to the Destructive Phase. The search process then "turns around" and proceeds toward (and past) the feasibility boundary by dropping variables. Eventually, we switch again to the Constructive Phase, where we turn around and move toward infeasibility by adding variables. It is at these *turn around* points that we use the foregoing memory to impel the search process to head in new directions in pursuit of new critical solutions. We sketch our rationale for accomplishing this as follows.

Suppose we have just switched from the Destructive Phase to the Constructive Phase. Our goal is to add variables that have not appeared at a value of 1 in recent critical solutions. Thus, we seek to impose the condition that the first variable added back, x_j, will have TABU_R(j) = 0. That is, we may conceive the condition TABU_R(j) > 0 as implying that x_j is tabu. However, this is not broad enough for our purpose. More generally, we seek to require that the first k variables added back (after turning around) will have TABU_R(j) = 0. Such a requirement may not be strictly possible. Consequently, we attach a large penalty weight, PEN_R, to TABU_R(j) and create a penalty value PEN_R*TABU_R(j). This penalty value is subtracted from the ratio evaluation, indicated earlier whose maximum value is used to choose the preferred move.

Likewise, frequency information is included by subtracting another penalty term, PEN_F*TABU_F(j), from the ratio evaluation. The positive weight, PEN_F, is scaled by the iteration count so that the penalty influence derived from long term frequency information is small compared to that of the short term recency information. In this manner, long term tabu information plays a subtle but useful role of breaking ties that may otherwise occur if one utilizes only short term tabu information.

Similarly, when we switch from the Constructive to the Destructive Phase, we seek to restrict our choice of variables to drop so that the first k variables chosen are selected from those x_j with maximum entries in the tabu lists TABU_R and TABU_F. Here, the tabu restriction is implemented by creating an inducement to drop a variable by subtracting an associated penalty term from the ratio evaluation and seeking the minimum penalized ratio.

Since the local search is based on a ratio evaluation, we choose the penalty coefficients by reference to this evaluation. This is done as follows. Each original constraint is scaled by its RHS value and the resulting weighted rows are summed to yield a surrogate constraint from which the ratio r_j , for each column j, is computed. Denote the maximum r_j ratio by r*. Then, our penalties are set as follows:

PEN_R = r*

PEN_F = r*/s

where s is a scale factor given by the product of the iteration count and a large, positive constant(C). Note that since the iteration count is an over estimate of the maximum TABU_F value at any iteration, the above calculation for the long term tabu penalty corresponds to scaling TABU_F by a simple function of the maximum TABU_F value. We experimented with several values (over a wide range) for C. For the results reported later in this paper, C was set to 100,000.

Other approaches to incorporating long term (e.g. TABU_F) information into the search process are available as well. For example, the penalties could be adaptively computed on the basis of the current surrogate constraint instead of relative to an initial surrogate constraint.

Also, TABU_F (j) could be normalized by the sum of the TABU_F(j) values. We comment briefly on this alternative, along with the notion of postponing the use of long term information, later in the paper.

2.2 Penalty Calculations.

The calculations for handling the penalty terms in both the Constructive and Destructive phases are as follows.

Denote the usual evaluation for variable x_j by ratio(j). Let count_var denote the number of variables chosen since the last turn-around and let j* be the index of the variable to be chosen next. Then in the Constructive Phase, we choose j* (from the set of all j such that x_j = 0) as follows:

If count_var > k, let value(j) = ratio(j)

if count_var ≤ = k, let value(j) = ratio(j) - PEN_R*TABU_R(j)
- PEN_F*TABU_F(j)

Then j* corresponds to the variable with the maximum value of value(j). Similarly, in the Destructive Phase, we choose j* (from the set of all j such that x_j = 1) in order to minimize value(j), using the same two calculations of this value shown above.

The parameter k, the number of tabu-influenced adds or drops to be made immediately after a "turn around" is managed in a fashion that fosters additional diversity in the search process. We start with k = 1, and after a number of iterations (e.g., 2t), we set k = k+1. We continue in this fashion until k reaches a limit (KMAX) at which point we set k back to 1 and the process repeats. In general, KMAX is a function of problem size. However, over a rather wide range of problems, KMAX = 4 has performed well in our testing.

2.3 Controlling the Outer Oscillation parameter (Span).

The oscillations about the feasibility boundary are shaped by the parameter *span*. Span is fixed for a certain number of iterations and then changed in a systematic fashion. In our implementation, we manage span in the Transfer Phase by the following rules:

Initialization:
Set span=1, choose values of parameters p1 and p2, and designate that span is increasing.

Transfer Phase (increasing span):

For span from 1 to p1: Allow p2*span executions of the Constructive and Destructive Phase and then increase span by 1.

For Span from p1+1 to p2: Allow p2 executions of the Constructive and Destructive Phases and then increase span by 1. When span is increased beyond p2, set it back to p2 and designate that span is decreasing.

Transfer Phase (decreasing span):

For span from p2 to p1+1: Allow p2 executions of the Constructive and Destructive Phases, and then decrease span by 1.

For span from p1 to 1: Allow p2*span executions of the Constructive and Destructive Phases, and then decrease span by 1. When span reaches 0, set it back to 1 and designate that span in increasing.

Large values of p1 and p2 foster aggressive diversification while smaller values facilitate a search in a closer neighborhood of the most recent critical solution. The default values we employ are p1 = 3 and p2 = 7. These particular values are supported by the fact that span values of 1 to 3, with a predominant emphasis on the value 3, have been found effective in the context of labor scheduling problems. We include values of span larger than p1 = 3 in order to expand the range of alternatives examined. Clearly other values for these parameters, and other schemes for managing span may hold promise as well. We undertook to implement easily determined small values, without spending a lot of time to explore alternatives. As it turned out, our simple parameter choices worked well for us, as shown in the following section on computational results.

3. Computational Experience.

Multidimensional knapsack problems have been the object of many research initiatives over the past forty years. Early work, exemplified by that of Lorie and Savage [13], was motivated by applications in capital budgeting. Since that time, many other applications have been discussed and a variety of algorithms have been proposed. This class of problems is known to be NP-hard, and despite the considerable attention these problems have received over the years, optimal solutions remain elusive.

Many researchers have attempted to locate high quality, if not optimal, solutions to MK problems by designing and employing various heuristics (e.g., Senju and Toyoda [15], Loulou and Michaelides [14], and Fréville and Plateau [5]). These approaches showed considerable promise and laid the groundwork for continuing work in designing heuristics for this class of problems.

Building on these foundations, recent papers by Drexl [4], Dammeyer and Voss [3], Aboudi and Jornsten [1] , and Glover and Lokketangen [12] have reported computational experience with new heuristic approaches to 57 standard multidimensional knapsack problems taken from a test set due to Drexel [4] which is currently available from Beasley [2]. Since this particular problem set has become the standard by which various approaches are compared, we chose to test our Tabu Search approach on these problems as well.

Starting with initial values (p_1 =3, p_2 =7, and t =7), each of the 57 standard test problems was run for a total of 50 outer SPAN oscillations. For each run (restart), KMAX was set to four. On 42 of the 57 problems, the optimal solution was found on the first run. On the other fifteen problems, optimal solutions were found on runs later than the first. For these runs, the initial parameter values were successively modified as follows. Keeping p_2 =7, t was decreased to 5 and then to 3. Parameter p_2 was then lowered to 5 and t was again allowed to take on the values 7, 5 and 3. In all additional runs, p_1 was kept at the default value of 3. Optimal solutions were found for all of the 57 test problems by this approach, which introduces a maximum of six restarts, each beginning from the same zero solution vector.

None of the other published approaches to these standard test problems report finding the optimal solution for all 57 problems. Drexl [4] obtained his results using two different implementations of a simulated annealing algorithm. Each problem was solved starting from 10 different starting points. That is, each problem was solved 20 times. He reports finding an optimal solution in 25 of the 57 problems tested.

The results reported by Dammeyer and Voss [3] were obtained from three different versions of an add/drop exchange heuristic with *reverse elimination* tabu lists. For each version, each problem was solved from 20 different starting points. The solutions reported are the best solutions found in the 60 attempts for each problem. Across the 57 test problems, they report an optimal solution in 41 cases.

Aboundi and Jornsten [1] report results obtained from their implementation of an LP-based "pivot and complement" heuristic, which they also guide by a form of tabu search. Twenty different versions (variations) of their heuristic were tried on each problem. The results they report are the best solutions found for each problem. Out of the 57 test problems they reported the optimal solution in 49 cases.

(However, no single variation they implemented performed nearly this well.)

Lokketangen and Glover [12] report a tabu search heuristic that also uses adjacent extreme point moves, which correspond to non-tabu pivots to adjacent basic feasible solutions. Each problem was solved twice (using a different move evaluation and choice rule). The results they report are the best solutions found for each problem. For the 57 problems, they report the optimal solution in 54 cases.

Comparisons of the various methods should be made with considerable care. Issues of tuning, coding, quality and number of starting points, and parameter settings make comparisons difficult. In addition, the approach of Glover and Lokketangen [12], is designed for general 0/1 mixed integer problems rather than being a specialized algorithm for multidimensional knapsack problems.

Further Remarks on our Computational testing:

1. The results we report were obtained without the use of any reductions or variable pegging due to pre-processing. The starting solution for each problem was $x = 0$.

2. The importance of generating additional trial solutions by a simple adjustment of variables (that overrides tabu conditions at critical events) is substantial. For the 57 problems reported on here, the optimal solution was obtained from such trial solutions about 82% of the time. This highlights the effectiveness of our search process in generating critical events in the neighborhood of the optimal solution.

3. Several problems required multiple span cycles (complete outer oscillation on both sides of the feasibility boundary) to reach an optimal solution. This highlights the difficulty of these

problems and underscores the effectiveness of our flexible tabu structure and strategic oscillation scheme in mounting a robust search of the solution space. Over the 57 problems, optimal solutions were found in an average of approximately 7 span cycles. Thirty nine of the problems were solved optimally in four or fewer cycles. Four of the problems took more than 25 cycles to locate the optimal solution and one problem took 40 cycles.

4. Many of the 57 problems were also solved optimally utilizing only short term information, that is, by setting PEN_F =0. Generally, long term frequency information appears most useful (leading to optimal solutions in fewer span cycles) on the more difficult problems.

In an effort to explore this further, along with some alternative ways of utilizing long term information, we conducted a variety of additional runs on the 10 most difficult problems. These problems required the largest number of full span cycles to locate optimal solution. Recall that the results reported above were obtained utilizing long term frequency information throughout the search process in the manner depicted in section 2.1. For the purpose of comparison, we solved these 10 problems again under the following conditions: (a) using short term memory only; (b) using long term memory, but suppressing (withholding) the influence of long term memory in the search process for a certain number of iterations; (c) incorporating long term information as a penalty by normalizing TABU_F(j) by the sum of the TABU_(k) values rather than the scheme outlined in section 2.1; and (d) using long term information as described in (c) but suppressing it's impact for a certain number of iterations. For alternatives (b) and (d), long term frequency information was withheld for 100 iterations. In all cases, exactly the same parameter values were used as those that led to the reference results.

The results obtained from these runs underscore the important role of long term information in the search process. On average, it took 2.25 times as many span cycles to locate the optimal solution when using short term information only. When using long term information, but withholding its influence for the first 100 iterations, (e.g. case (b)), it took an average of 1.875 times as many span cycles as the reference case. Finally, for the conditions of (c) and (d), it took, respectively, 3.125 and 3.375 times as many span cycles on average to locate the optimal as the reference case.

Thus, on the problems considered here, the implementation described in this paper outperformed alternatives (a), (b), (c), and (d) in terms of <u>average</u> performance by a substantial amount. In addition, it produced the individual best performance on 6 out of the 10 problems and was generally close to most preferred on the others. Alternatives (a), (b) and (d) each produced the best individual problem result for one of the ten problems and there was a tie for best performance on the tenth problem.

5. In addition to the standard 57 test problems, we tested our approach on 24 randomly generated (correlated) problems ranging in size up to 500 variables and 25 constraints. Where possible, optimal solutions were obtained by a branch and bound algorithm in order to assess the performance of our heuristic. These problems are fairly difficult and in some cases the branch and bound algorithm ran for more than 4 consecutive days on a 486 machine without terminating with a proven optimal solution. For 23 of these 24 problems, our approach generated solutions that were generally appreciably superior to those given by the branch and bound algorithm. In

the one exception, we reported an objective function value of 4524 compared to the branch and bound generated value of 4525. In all cases, our procedure required a fraction of the time taken by the branch and bound procedure, completing its run in about 28 CPU minutes on average, and 355 CPU minutes in the worst case. (This represents a speed difference of about 50 to 1.) To provide an additional (perhaps more informative) basis of comparison, we note that the time required to *first* reach the best solution found by each of these methods averaged 22 CPU minutes for our TS method and 1250 CPU minutes for the branch and bound (yielding a time advantage for the TS approach of about 55 to 1). The fact that TS additionally found better solutions in 23 of the 24 cases increases the significance of these differences.

4. Conclusion.

In this paper we report an implementation of a new tabu search approach to multidimensional knapsack problems. Our approach employs a flexible memory structure that integrates recency and frequency information keyed to critical events of the search process. The method is enhanced by a strategic oscillation scheme that alternates between constructive and destructive phases, and drives the search to variable depths on each side of the feasibility boundary.

Our approach successfully obtained optimal solutions for each of 57 standard test problems from the literature, a level of performance not matched by other attempts reported in the literature. Moreover, we found best known solutions to 23 of 24 additional multidimensional knapsack problems (correlated, randomly generated). Our reliance upon exceedingly simple parameter values, without conducting extensive exploration of alternatives, suggests that the use of such critical event memory in tabu search may be valuable in other applications.

5. REFERENCES

R.Aboudi and K. Jornsten (1994) "Tabu Search for General Zero Integer Programs Using the Pivot and Complement Heuristic," *ORSA Journal on Computing*, Vol. 6, No. 1, Winter 1994, pp. 82-93.

J. Beasley (1995) "OR-Library," available by anonymous ftp to msemga.ms.ic.ac.uk.

F. Dammeyer and S. Voss (1993) "Dynamic Tabu List Management Using Reverse Elimination Method," *Annals of Operations Research*, No. 41, pp. 31-46.

A. Drexl (1987) "A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem," *Computing*, No. 40, pp. 1-8.

A. Fréville and G. Plateau (1986) "Heuristics and Reduction Methods for Multiple Constraints 0-1 Linear Programming Problems," *European Journal of Operations Research*, No. 24, pp. 206-215.

F. Glover (1965) "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," *Operations Research,* 13, 879-919.

F. Glover (1977) "Heuristics in Integer Programming Using Surrogate Constraints," *Decision Sciences*, Vol. 8, No. 1, January, pp. 156-166.

F. Glover (1994) "Tabu Search Fundamentals and Uses," University of Colorado Working Paper.

F. Glover (1993) "Tabu Thresholding: Improved Search by Nonmonotonic Trajectories," to appear in *ORSA Journal on Computing.*

F. Glover and M. Laguna (1993) "Tabu Search," *Modern Heuristic Techniques for Combinatorial Problems*, C. Reeves, ed., Blackwell Scientific Publishing, pp. 70-141.

F. Glover, M. Laguna, E. Taillard, and D. de Werra, eds (1993) "Tabu Search," special issues of the *Annals of Operations Research*, Vol. 41, J.C. Baltzer.

F. Glover and A. Lokketangen (1994) "Solving Zero-One Mixed Programming Problems Using Tabu Search," University of Colorado Working Paper.

J. Lorie and L. Savage (1955) "Three Problems in Capital Rationing," *Journal of Business*, No. 28, pp. 229-239.

R. Loulou and E. Michaelides (1979) "New Greedy-Like Heuristics for the Multidimensional 0-1 Knapsack Problem," *Operations Research*, no. 27, pp. 1101-1114.

S. Senju, and Y. Toyoda (1968) "An Approach to Linear Programming With 0-1 Variables," *Management Science*, No. 15., pp. 196-207.