Chapter 3

Tabu Search

Fred Glover and Manuel Laguna

3.1 Introduction

Tabu search (TS) has its antecedents in methods designed to cross boundaries of feasibility or local optimality normally treated as barriers, and systematically to impose and release constraints to permit exploration of otherwise forbidden regions. Early examples of such procedures include heuristics based on surrogate constraint methods and cutting plane approaches that systematically violate feasibility conditions. The modern form of tabu search derives from Glover [1]. Seminal ideas of the method are also developed by Hansen [2] in a *steepest ascent/mildest descent* formulation. Additional contributions, such as those cited in the following pages, are shaping the evolution of the method and are responsible for its growing body of successful applications.

Webster's dictionary defines *tabu* or *taboo* as 'set apart as charged with a dangerous supernatural power and forbidden to profane use or contact...' or 'banned on grounds of morality or taste or as constituting a risk...'. Tabu search scarcely involves reference to supernatural or moral considerations, but instead is concerned with imposing restrictions to guide a search process to negotiate otherwise difficult regions. These restrictions operate in several forms, both by direct exclusion of certain search alternatives classed as 'forbidden', and also by translation into modified evaluations and probabilities of selection.

The purpose of this chapter is to integrate some of the fundamental ways of viewing and characterizing tabu search, with extended

examples to clarify its operations. We also point to a variety of directions for new applications and research. Our development includes comparisons and contrasts between the principles of tabu search and those of simulated annealing (SA) and genetic algorithms (GAs). Computational implications of these differences, and foundations for creating hybrid methods that unite features of these different approaches are also discussed. In addition, we examine special designs and computational outcomes for incorporating tabu search as a driving mechanism within neural networks.

The philosophy of tabu search is to derive and exploit a collection of principles of intelligent problem solving. A fundamental element underlying tabu search is the use of flexible memory. From the standpoint of tabu search, flexible memory embodies the dual processes of creating and exploiting structures for taking advantage of history (hence combining the activities of acquiring and profiting from information).

The memory structures of tabu search operate by reference to four principal dimensions, consisting of recency, frequency, quality, and influence. These dimensions in turn are set against a background of logical structure and connectivity. The rôle of these elements in creating effective problem-solving processes provides the focus of our following development.

3.2 The Tabu Search Framework

To provide a background for understanding some of the fundamental elements of tabu search, we illustrate its basic operation with an example.

3.2.1 An illustrative example

Permutation problems form an important class of problems in optimization, and offer a useful vehicle to demonstrate some of the considerations that must be faced in the combinatorial domain. Classical instances of permutation problems include the travelling salesman problem, the quadratic assignment problem, production sequencing problems, and a variety of design problems. As a basis for illustration, consider the problem of designing a material consisting of a number of insulating modules. The order in which these modules are

arranged determines the overall insulating property of the resulting material, as shown in Figure 3.1.

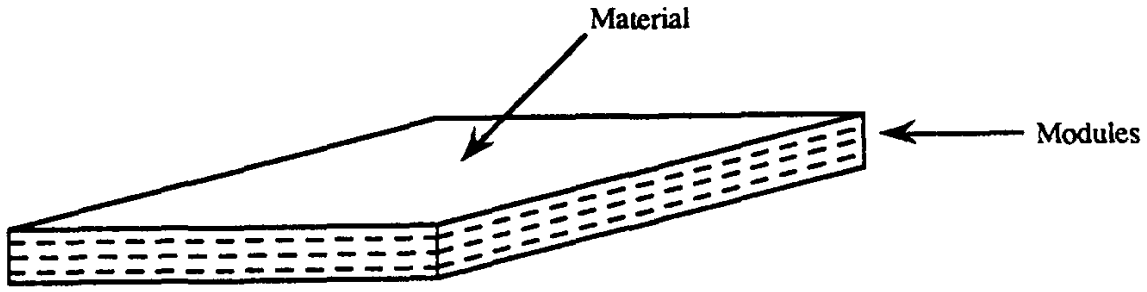


Figure 3.1: Modules in an insulating material

The problem is to find the ordering of modules that maximizes the overall insulating property of the composite material. Suppose that 7 modules are considered for a particular material, and that evaluating the overall insulating property of a particular ordering is a computationally expensive procedure. We desire a search method that is able to find an optimal or near-optimal solution by examining only a small subset of the total number of permutations possible (in this case 5040, though for many applications it can be astronomical).

Closely related problems that can be represented in essentially the same way include serial filtering and job sequencing problems. Serial filtering problems arise in pattern recognition and signal processing applications, where a given input is to be subjected to a succession of filters (or screening tests) to obtain the 'best' output. Filters are sequentially applied to the input signal, and the quality of the output is determined by the order in which they are placed (see Figure 3.2). In this case, the search method must be designed to find the best filtering sequence. Such filtering processes are also relevant to applications in chemical engineering, astronomy, and biochemistry.

Job sequencing problems consist of determining best sequences for processing a set of jobs on designated machines. Each machine is thus assigned some permutation of available jobs. (In some settings, multiple machine problems may be treated by extensions of processes

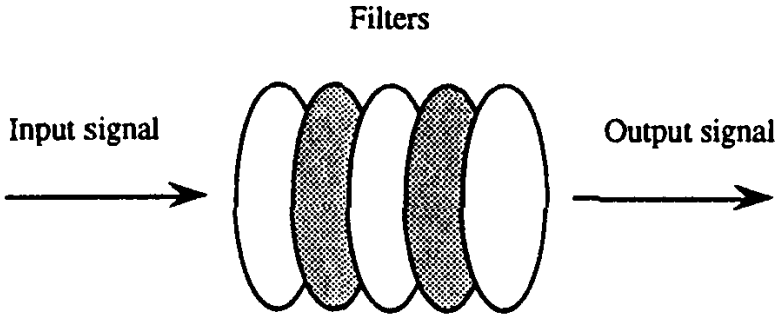


Figure 3.2: Filtering sequence

for single machine problems.) There are many variants of the single machine problem depending on the definition of 'best' sequence. For example, the best sequence may be the one that minimizes the *makespan*—*the completion time of the last job in the sequence. Other* possibilities are to minimize a weighted sum of tardiness penalties or a sum of setup costs.

For well-structured objective functions, evaluations of ways to move from one solution to another are generally fast. However, problems with even modest numbers of jobs overwhelm the capabilities of algorithms that 'guarantee' optimality, rendering them unable to obtain solutions in reasonable amounts of time. That is one of the reasons why effective heuristic approaches have proved important in the area of production scheduling.

Some useful variants of the foregoing problems can be represented 'as if' they were permutation problems. These include, for example, problems where it is simultaneously desired to select a best subset of items (modules, filters, jobs) from an available pool, and to identify a best sequence for this chosen set. In this case, the problem can be represented by creating a dummy position to hold a residual pool, where all items that do not currently occupy one of the sequence positions are placed. (The path assignment problem discussed in Section 3.4 is a good example of this kind of representation.)

We focus on the module insulation problem, using it to introduce and illustrate the basic components of tabu search. First we assume that an initial solution for this problem can be constructed in some

intelligent fashion, i.e. by taking advantage of some problem-specific structure. Suppose the initial solution to our problem is the one shown in Figure 3.3.

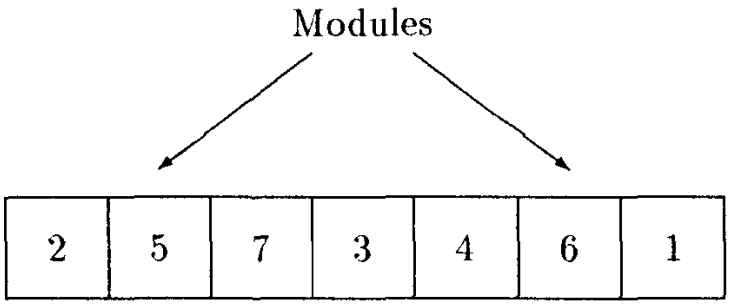


Figure 3.3: Initial permutation

The ordering in Figure 3.3 specifies that module 2 is placed in the first position, followed by module 5, etc. The resulting material has an insulating property of 10 units (which we assume was found by an accompanying evaluation routine, e.g. a simulator package for estimating the properties of a material without actually building a prototype). TS methods operate under the assumption that a *neighbourhood* can be constructed to identify 'adjacent solutions' that can be reached from any current solution. (Neighbourhood search is described in Section 3.2.3.) Pairwise exchanges (or swaps) are frequently used to define neighbourhoods in permutation problems, identifying *moves* that lead from one solution to the next. In our problem, a swap exchanges the position of two modules as illustrated in Figure 3.4. Therefore, the complete neighbourhood of a given current solution consists of the 21 adjacent solutions that can be obtained by such swaps.

Associated with each swap is a move value, which represents the change in the objective function value as a result of the proposed exchange. Move values generally provide a fundamental basis for evaluating the quality of a move, although other criteria can also be important, as indicated later. A chief mechanism for exploiting memory in tabu search is to classify a subset of the moves in a neighbourhood as forbidden (or tabu). The classification depends on the history of the search, particularly as manifested in the recency or frequency that certain move or solution components, called *attributes*,

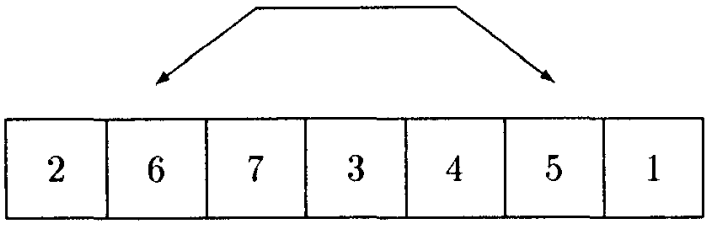


Figure 3.4: Swap of modules 5 and 6

have participated in generating past solutions. For example, one attribute of a swap is the identity of the pair of elements that change positions (in this case, the two modules exchanged). As a basis for preventing the search from repeating swap combinations tried in the recent past, potentially reversing the effects of previous moves by interchanges that might return to previous positions, we will classify as tabu all swaps composed of any of the most recent pairs of such modules; in this case, for illustrative purposes, the three most recent pairs. This means that a module pair will be kept tabu for a duration (tenure) of 3 iterations. Since exchanging modules 2 and 5 is the same as exchanging modules 5 and 2, both may be represented by the pair (2,5). Thus, a data structure such as the one shown in Figure 3.5 may be used.

Each cell of the structure in Figure 3.5 contains the number of iterations remaining until the corresponding modules are allowed to exchange positions again. Therefore, if the cell (3,5) has a value of zero, then modules 3 and 5 are free to exchange positions. On the other hand, if cell (2,4) has a value of 2, then modules 2 and 4 may not exchange positions for the next two iterations (i.e. a swap that exchanges these modules is classified tabu).

The type of move attributes illustrated here for defining tabu restrictions is not the only one possible. For example, reference may be made to separate modules rather than module pairs, or to positions of modules, or to links between their immediate predecessors (or successors), and so forth. Some choices of attributes are better than others, and relevant considerations are discussed in Section 3.2.5. (Attributes involving created and broken links between immediate predecessors and successors are often among the more effective for many permutation problems.)

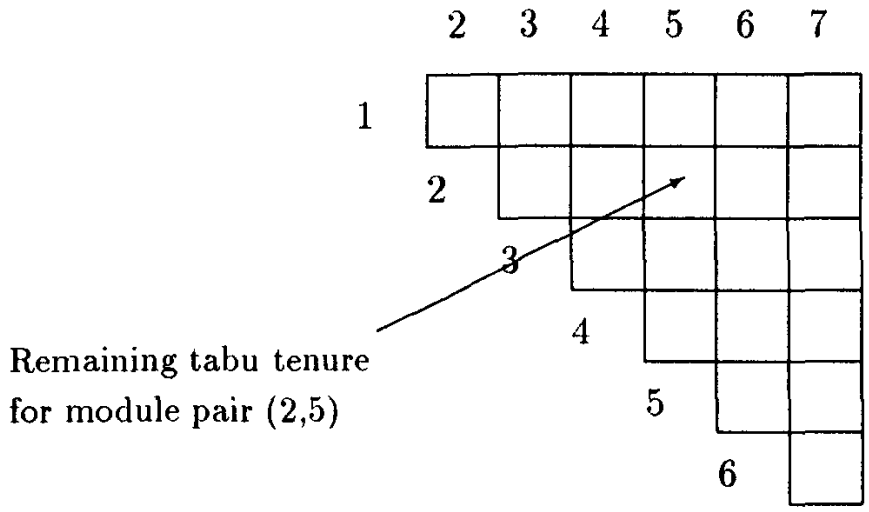


Figure 3.5: Tabu data structure for attributes consisting of module pairs exchanged

To implement tabu restrictions such as those based on module pairs, an important exception must be taken into account. Tabu restrictions are not inviolable under all circumstances. When a tabu move would result in a solution better than any visited so far, its tabu classification may be overridden. A condition that allows such an override to occur is called an *aspiration criterion.* (Several useful forms of such criteria are presented in Section 3.2.7.) The following shows 4 iterations of the basic tabu procedure that employs the *paired module* tabu restriction and the *best solution* aspiration criterion.

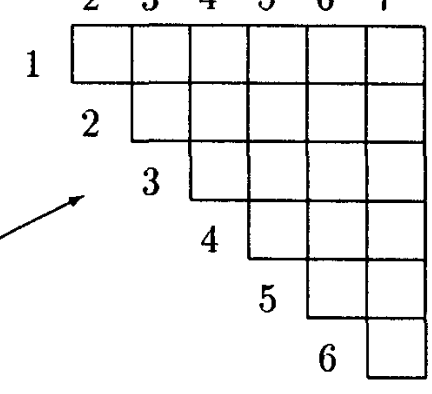
Iteration 0 (*Starting point*)

Current solution

2	5	7	3	4	6	1

Insulation Value=10

Tabu structure



Top 5 candidates

Swap	Value	
5,4	6	*
7,4	4	
3,6	2	
2,3	0	
4,1	-1	

The starting solution has an insulation value of 10, and the tabu data structure is initially empty, i.e. it is filled with zeros, indicating no moves are classified tabu at the beginning of the search. (For clarity, we have not actually inserted these zeros in this or the following diagrams.) After evaluating the candidate swap moves, the top five moves (in terms of move values) are shown in the table for iteration 0 above. This information is provided by an independent evaluation subroutine designed to identify move values for this particular problem. (Of course, it is not necessary for the subroutine to sort and identify each of the 5 best moves, since we are interested only in the best. The additional options are included here to clarify certain ideas subsequently presented.) To find a local maximum for the insulating property of the material, we swap the positions of modules 5 and 4 (as indicated by the asterisk). The total gain of such a move equals 6 units.

Iteration 1

Current solution

2	4	7	3	5	6	1

Insulation Value=16

Tabu structure

	2	3	4	5	6	7
1						
2						
3						
4				3		
5						
6						

Top 5 candidates

Swap	Value	
3,1	2	*
2,3	1	
3,6	-1	
7,1	-2	
6,1	-4	

The new current solution has an insulating value of 16 (i.e. the previous insulation value plus the value of the selected move). The tabu structure now shows that swapping the positions of modules 4 and 5 is forbidden for 3 iterations. The most improving move at this step is to swap 3 and 1 for a gain of 2.

Iteration 2

Current solution

2	4	7	1	5	6	3

Insulation Value=18

Tabu structure

	2	3	4	5	6	7
1		3				
2						
3						
4				2		
5						
6						

Top 5 candidates

Swap	Value	
1,3	-2	T
2,4	-4	*
7,6	-6	
4,5	-7	T
5,3	-9	

The new current solution becomes the best solution found so far with an insulating value of 18. At this iteration, two exchanges are classified tabu, as indicated by the nonzero entries in the tabu structure.

Note that entry (4,5) has been decreased from 3 to 2, indicating that its original tabu tenure of 3 now has 2 remaining iterations to go. This time, none of the candidates (including the top 5 shown) has a positive move value. Therefore, a non-improving move has to be made. The most attractive non-improving move is the reversal of the move performed in the previous iteration, but since it is classified tabu, this move is not selected. Instead, the swap of modules 2 and 4 is chosen, as indicated by the asterisk.

Iteration 3

Current solution

4	2	7	1	5	6	3

Insulation Value=14

Tabu structure

	2	3	4	5	6	7
1		2				
2			3			
3						
4				1		
5						
6						

Top 5 candidates

Swap	Value	
4,5	6	T*
5,3	2	
7,1	0	
1,3	-3	T
2,6	-6	

The new current solution has an insulation value inferior to the two values previously obtained, as a result of executing a move with a negative move value. The tabu data structure now indicates that 3

moves are classified tabu, with different remaining tabu tenures. At the top of the candidate list, we find the swap of modules 4 and 5, which in effect represents the reversal of the first move performed, and is classified tabu. However, performing this move produces a solution with an objective function value that is superior to any previous insulation value. Therefore, we make use of the aspiration criterion to override the tabu classification of this move and select it as the best on this iteration.

Iteration 4

Current solution

5	2	7	1	4	6	3

Insulation Value=20

Tabu structure

	2	3	4	5	6	7
1		1				
2			2			
3						
4				3		
5						
6						

Top 5 candidates

Swap	Value	
7,1	0	*
4,3	-3	
6,3	-5	
5,4	-6	T
2,6	-8	

The current solution becomes the incumbent new best solution and the process continues. Note that the chosen tabu restriction and tabu tenure of 3 results in forbidding only 3 out of 21 possible swaps, since the module pair with a residual tenure of 1 always drops to a residual tenure of 0 each time a new pair with tenure 3 is introduced. (By recording the iteration when a module pair becomes tabu, and comparing this against the current iteration to determine the remaining tabu tenure, it is unnecessary to change these entries at each step as we do here.)

In some situations, it may be desirable to increase the percentage of available moves that receive a tabu classification. This may be achieved either by increasing the tabu tenure or by changing the tabu restriction. For example, a tabu restriction that forbids swaps containing at least one member of a module pair will prevent a larger number of moves from being executed, even if the tenure remains the same. (In our case, this restriction would forbid 15 out of 21 swaps if the tabu tenure remains at 3.) Such a restriction is based

on single module attributes instead of paired module attributes, and can be implemented with much less memory, i.e. by an array that records a tabu tenure for each module separately. Generally speaking, regardless of the type of restriction selected, improved outcomes are often obtained by tabu tenures that vary dynamically, as described in Section 3.2.6.

Move Values and Updates Because tabu search aggressively selects best admissible moves (where the meaning of best is affected by tabu classification and other elements to be indicated), it must examine and compare a number of move options. For many problems, only a portion of the move values will change from one iteration to the next, and often these changed values can be isolated and updated very quickly. For example, in the present illustration it may be useful to store a table $move\_value(j, k)$, which records the current move value for exchanging modules j and k. When a move is executed, a relatively small part of this table (consisting of values that change) can be quickly modified, and the updated table can then be consulted to identify moves that become the new top candidates.

Such partial updating often can be further enhanced by a list $move\_name(move\_value)$ which, for each *move_value* in a relevant range, identifies *move_name* to be a specific move that yields this value. A linked list then can connect this *move_name* to the names of all other moves that yield the same *move_value*. The combination of the $move\_name(move\_value)$ array and the linked list can be updated very quickly to make it easy to locate moves with best move values in cases where only a relatively small number of elements change. A given *move_value* entry can also refer to a range of move values, with an option to regard all values within a specified range as 'essentially equivalent'. (However, we suggest the merit of differentiating members of a given range more carefully upon approaching local optimality.)

On a broader scale, lists to facilitate access to best moves invite differentiation to include considerations introduced by move influence (Section 3.2.7) and by candidate list strategies (Section 3.3). They also are subject to periodic scanning with reference to concerns that extend beyond the short term horizon, as we illustrate next.

Complementary Tabu Memory Structures The accompaniment of recency-based memory with frequency-based memory adds a component that typically operates over a longer horizon. To illustrate one of the useful longer term applications of frequency-based memory, suppose that 25 TS iterations have been performed, and that the number of times each module pair has been exchanged is saved in an expanded tabu data structure. The lower diagonal of this structure now contains the frequency counts.

Iteration 26

Current solution

1	3	6	2	7	5	4

Insulation Value=12

Tabu structure (Recency)

	1	2	3	4	5	6	7
1				3			
2							
3	3					2	
4	1	5					1
5		4		4			
6			1		2		
7	2			3			

(Frequency)

Top 5 candidates

Swap	Value	Penalized Value	
1,4	3	3	T
2,4	-1	-6	
3,7	-3	-3	*
1,6	-5	-5	
6,5	-4	-6	

At the current iteration (iteration 26), the recency memory indicates that the last three module pairs exchanged were (1,4), (3,6), and (4,7). The frequency counts show the distribution of moves throughout the first 25 iterations. We use these counts to *diversify* the search, driving it into new regions. This diversifying influence is restricted to operate only on particular occasions. In this case, we select those occasions where no admissible improving moves exist. Our use of the frequency information will penalize non-improving moves by assigning a larger penalty to swaps of module pairs with greater frequency counts. (Typically these counts would be normalized, as by dividing by the total number of iterations or their maximum values.) We illustrate this in the present example by simply subtracting a frequency count from the associated move value.

The list of top candidates for iteration 26 shows that the most improving move is the swap (1,4), but since this module pair has a residual tabu tenure of 3, it is classified tabu. The move (2,4) has a

value of -1, and it might otherwise be the one next preferred, except that its associated modules have been exchanged frequently during the history of the search (in fact, more frequently than any other module pair). Therefore, the move is heavily penalized and it loses its attractiveness. The swap of modules 3 and 7 is thus selected as the best move on the current iteration.

The strategy of instituting penalties only under particular conditions is used to preserve the aggressiveness of the search. Penalty functions in general are designed to account not only for frequencies but also for move values and certain influence measures, as discussed in Section 3.2.8.

In addition, frequencies defined over different subsets of past solutions, particularly subsets of élite solutions consisting of high quality local optima, give rise to complementary strategies of *intensification*. Intensification and diversification strategies interact to provide fundamental cornerstones of longer term memory in tabu search. The ways in which such elements are capable of creating enhanced search methods, extending the simplified approach of the preceding example, are elaborated in following sections.

3.2.2 Notation and problem description

A few basic definitions and conventions are useful as a foundation for communicating the principal ideas of TS. For this purpose we express the mathematical optimization problem in a slightly more general form than that used in chapter 1.

$$\begin{aligned} &\textit{Minimise} && c(x) \\ &\textit{subject to} && x \in \mathbf{X} \end{aligned}$$

The objective function $c(x)$ may be linear or nonlinear, and the condition $x \in \mathbf{X}$ summarizes constraints on the vector x. These constraints may include linear or nonlinear inequalities (as in chapter 1), and may compel some or all components of x to receive discrete values.

In many applications of combinatorial optimization, the problem of interest is not explicitly formulated as we have shown it. In such cases the present formulation may be conceived as a code for another formulation. The requirement $x \in \mathbf{X}$, for example, may specify

logical conditions or interconnections that would be cumbersome to formulate mathematically, but may better be left as verbal stipulations (for example, in the form of rules). Often in these instances the variables are simply codes for conditions or assignments that are parts of the more complex structure. For example, an element of x may be a binary variable that receives a value of 1 to code for assigning an element u to a set or position v, and that receives a value 0 to indicate the assignment does not occur.

3.2.3 Neighbourhood search

TS may be conveniently characterized as a form of neighbourhood search, which has already been described in chapter 2. However, here we wish to define neighbourhood search in a less restricted fashion than usual. Frequently, for example, constructive and destructive procedures are excluded, whereas such procedures and their combinations are routinely subjected to the guidance of TS.

In neighbourhood search, each solution $x \in \mathbf{X}$ has an associated set of neighbours, $N(x) \subset \mathbf{X}$, called the neighbourhood of x. Each solution $x' \in N(x)$ can be reached directly from x by an operation called a *move*, and x is said to move (or transition) to x' when such an operation is performed. Normally in TS, neighbourhoods are assumed symmetric, i.e. x' is a neighbour of x if and only if x is a neighbour of x'.

Neighbourhood Search Method

Step 1	(Initialization)
(A)	Select a starting solution $x^{now} \in \mathbf{X}$.
(B)	Record the current best known solution by setting $x^{best} = x^{now}$ and define $best\_cost = c(x^{best})$.
Step 2	(Choice and termination) Choose a solution $x^{next} \in N(x^{now})$. If the choice criteria employed cannot be satisfied by any member of $N(x^{now})$ (hence no solution qualifies to be x^{next}), or if other termination criteria apply (such as a limit on the total number of iterations), then the method stops.
Step 3	(Update) Re-set $x^{now} = x^{next}$, and if $c(x^{now}) < best\_cost$, perform Step 1(B). Then return to Step 2.

The steps of neighbourhood search are as described above, where we assume choice criteria for selecting moves, and termination criteria for ending the search, are given by some external set of prescriptions.

The foregoing procedure can represent a constructive method by stipulating that **X** is expanded to include x vectors whose components take null (unassigned) values, and by stipulating that a neighbour x' of x can result by replacing a null component of x with a non-null component. (A change of representation sometimes conveniently allows null components to be represented by values of 0 and non-null components by values of 1.) A standard constructive method does not yield symmetric neighbourhoods, since non-null components are not permitted to become null again (hence the method ends when no more components are null). However, tabu search reinstates the symmetric relation by allowing constructive and destructive moves to co-exist, as a special instance of an approach called strategic oscillation (see Section 3.3).

The neighbourhood search method can easily be altered by adding special provisions to yield a variety of classical procedures. Descent methods, which only permit moves to neighbour solutions that improve the current $c(x^{now})$ value, and which end when no improving solutions can be found, can be expressed by the following provision in Step 2.

Descent Method

Step 2	(Choice and termination) Choose $x^{next} \in N(x^{now})$ to satisfy $c(x^{next}) < c(x^{now})$ and terminate if no such x^{next} can be found.

The evident shortcoming of a descent method is that the final x^{now} obtained is a local optimum, which in most cases will not be a global optimum.

Randomized procedures such as Monte Carlo methods, which include simulated annealing, can similarly be represented by adding a simple provision to Step 2.

Monte Carlo Method

Step 2	(Choice and termination)
(A)	Randomly select x^{next} from $N(x^{now})$.
(B)	If $c(x^{next}) \leq c(x^{now})$ accept x^{next} (and proceed to Step 3).
(C)	If $c(x^{next}) > c(x^{now})$ accept x^{next} with a probability that decreases with increases in the difference $c(x^{next}) - c(x^{now})$. If x^{next} is not accepted on the current trial by this criterion, return to (A).
(D)	Terminate by a chosen cutoff rule.

Monte Carlo methods continue to sample the search space until finally terminating by some form of iteration limit. Normally they use an exponential function to define probabilities, drawing from practice established in engineering and physical science. As described in chapter 2, the Monte Carlo version represented by simulated annealing starts with a high probability for accepting non-improving moves in Step 2(C) which is decreased over time as a function of a parameter called the 'temperature' which monotonically diminishes toward zero as the number of iterations grows.

Such approaches offer a chance to do better than finding a single local optimum since they effectively terminate only when the probability of accepting a non-improving move in Step 2(C) becomes so small that no such move is ever accepted (in the finite time allowed). Hence, they may wander in and out of various intermediate local optima prior to becoming lodged in a final local optimum, when the temperature becomes small.

Another randomizing approach to overcome the limitation of the descent method is simply to restart the method with different randomly selected initial solutions, and run the method multiple times. Such a random restart approach, sometimes called *iterated descent*, may be contrasted with a random perturbation approach, which simply chooses moves randomly for a period after reaching each local optimum, and then resumes a trajectory of descent. Alternating threshold methods indicated in Section 3.2.7 provide a refinement of this idea.

3.2.4 Tabu search characteristics

Tabu search, in contrast to the preceding methods, employs a somewhat different philosophy for going beyond the criterion of termi-

nating at a local optimum. Randomization is de-emphasized, and generally is employed only in a highly constrained way, on the assumption that intelligent search should be based on more systematic forms of guidance. Thus randomization (pseudo-randomization) is chiefly assigned the rôle of facilitating operations that are otherwise cumbersome to implement or whose strategic implications are unclear. (In the latter case, a supplementary learning approach such as target analysis—see Laguna and Glover [3]—is customarily employed to determine if such implications can be sharpened.) Accordingly, many tabu search implementations are largely or wholly deterministic. An exception occurs for the variant called probabilistic tabu search, which selects moves according to probabilities based on the status and evaluations assigned to these moves by the basic tabu search principles. (A discussion of probabilistic convergence issues is provided by Faigle and Kern [4].)

Special TS uses of memory: modifying neighbourhood structures

The notion of exploiting certain forms of flexible memory to control the search process is the central theme underlying tabu search. The effect of such memory may be envisioned by stipulating that TS maintains a selective history H of the states encountered during the search, and replaces $N(x^{now})$ by a modified neighbourhood which may be denoted $N(H, x^{now})$. History therefore determines which solutions may be reached by a move from the current solution, selecting x^{next} from $N(H, x^{now})$.

In TS strategies based on short term considerations, $N(H, x^{now})$ is typically a subset of $N(x^{now})$, and the tabu classification serves to identify elements of $N(x^{now})$ excluded from $N(H, x^{now})$. In the intermediate and longer term strategies, $N(H, x^{now})$ may contain solutions not in $N(x^{now})$, generally consisting of selected élite solutions (high quality local optima) encountered at various points in the solution process. Such élite solutions are typically identified as elements of a regional cluster in intermediate term intensification strategies, and as elements of different clusters in longer term diversification strategies. In addition, élite solution components, in contrast to the solutions themselves, are included among the elements that can be retained and integrated to provide inputs to the search process.

TS also uses history to create a modified evaluation of currently accessible solutions. This may be expressed formally by saying that TS replaces the objective function $c(x)$ by a function $c(H, x)$, which has the purpose of evaluating the relative quality of currently accessible solutions. (An illustration is provided by the use of frequency-based memory in the example of Section 3.2.1.) The relevance of this modified function occurs because TS uses aggressive choice criteria that seek a best x^{next}, i.e. one that yields a best value of $c(H, x^{next})$, over a candidate set drawn from $N(H, x^{now})$. Moreover, modified evaluations are often accompanied by systematic alteration of $N(H, x^{now})$, to include neighbouring solutions that do not satisfy customary feasibility conditions (i.e. that strictly speaking do not yield $x \in \mathbf{X}$). Reference to $c(x)$ is retained for determining whether a move is improving or leads to a new best solution.

For large problems, where $N(H, x^{now})$ may have many elements, or for problems where these elements may be costly to examine, the aggressive choice orientation of TS makes it highly important to isolate a candidate subset of the neighbourhood, and to examine this subset instead of the entire neighbourhood. This can be done in stages, allowing the candidate subset to be expanded if alternatives satisfying aspiration levels are not found. Because of the significance of the candidate subset's rôle, we refer to this subset explicitly by the notation $Candidate\_N(x^{now})$. Then the tabu search procedure may be expressed in the following manner.

Tabu Search Method

Step 1	(Initialization) Begin with the same initialization used by Neighbourhood Search, and with the history record H empty.
Step 2	(Choice and termination) Determine $Candidate\_N(x^{now})$ as a subset of $N(H, x^{now})$. Select x^{next} from $Candidate\_N(x^{now})$ to minimize $c(H, x)$ over this set. (x^{next} is called a highest evaluation element of $Candidate\_N(x^{now})$.) Terminate by a chosen iteration cut-off rule.
Step 3	(Update) Perform the update for the Neighbourhood Search Method, and additionally update the history record H.

Formally the tabu search method is quite straightforward to state. The essence of the method depends on how the history record H is

defined and used, and on how the neighbourhood *Candidate_N*(x^{now}) and the evaluation function $c(H, x)$ are determined.

In the simplest cases we may imagine *Candidate_N*(x^{now}) to constitute all of $N(H, x^{now})$, and take $c(H, x) = c(x)$, disregarding neighbourhood screening approaches and the longer term considerations that introduce élite solutions into the determination of moves. We begin from this point of view, focusing on the short term component of tabu search for determining the form and use of H. The basic considerations provide a foundation for the intermediate and long term TS components as well.

3.2.5 Tabu search memory

Attribute based memory

An attribute of a move from x^{now} to x^{next}, or more generally of a trial move from x^{now} to a tentative solution x^{trial}, can encompass any aspect that changes as a result of the move. Natural types of attributes are as follows.

Illustrative Move Attributes for a Move x^{now} to x^{trial}

(A1)	Change of a selected variable x_j from 0 to 1.
(A2)	Change of a selected variable x_k from 1 to 0.
(A3)	The combined change of (A1) and (A2) taken together.
(A4)	Change of $c(x^{now})$ to $c(x^{trial})$.
(A5)	Change of a function $g(x^{now})$ to $g(x^{trial})$ (where g may represent a function that occurs naturally in the problem formulation or that is created strategically).
(A6)	Change represented by the difference value $g(x^{trial}) - g(x^{now})$.
(A7)	The combined changes of (A5) or (A6) for more than one function g considered simultaneously.

A single move can evidently give rise to multiple attributes. For example, a move that changes the values of two variables simultaneously may give rise to each of the three attributes (A1), (A2), and (A3), as well as to other attributes of the form indicated. Attributes that represent combinations of other attributes do not necessarily provide more exploitable information, as will be seen. Attributes (A5) to (A7) are based on a function g that may be strategically chosen to be completely independent from c. For example, g may be a measure of distance (or dissimilarity) between any given solution and a reference

solution, such as the last local optimum visited or the best solution found so far. Then, attribute (A6) would indicate whether a trial solution leads the search further from or closer to the reference point.

Move attributes, involving change, may be subdivided into component attributes called *from-attributes* and *to-attributes*. That is, each move attribute may be expressed as an ordered pair (*from-attribute, to-attribute*) whose components are respectively attributes of the solutions x^{now} and x^{trial}. Letting $A(x^{now})$ and $A(x^{trial})$ denote attribute sets for these two solutions, the requirement of change underlying the definition of a move attribute implies

$$\begin{aligned} \textit{from-attribute} &\in A(x^{now}) - A(x^{trial}) \\ \textit{to-attribute} &\in A(x^{trial}) - A(x^{now}). \end{aligned}$$

This differentiation between move attributes and their component *from-attributes* and *to-attributes* is useful for establishing certain outcomes related to their use.

When we refer to assigning alternative values to a selected variable x_j of x, and particularly to assigning values 0 and 1 to a binary variable, we will understand by our previous conventions that this can refer to a variety of operations such as adding or deleting edges from a graph, assigning or removing a facility from a particular location, changing the processing position of a job on a machine, and so forth. Such coding conventions can be extended to include the creation of supplementary variables that represent states of subservient processes. For example, $x_j = 0$ or 1 may indicate that an associated variable is nonbasic or basic in an extreme point solution procedure, as in the simplex method and its variants for linear and nonlinear programming.

Uses of move attributes

Recorded move attributes are often used in tabu search to impose constraints, called tabu restrictions, that prevent moves from being chosen that would reverse the changes represented by these attributes. More precisely, when a move from x^{now} to x^{next} is performed that contains an attribute e, a record is maintained for the reverse attribute which we denote by $\bar{e}$, in order to prevent a move from occurring that contains some subset of such reverse attributes. Examples of kinds of tabu restrictions frequently employed are as follows.

Illustrative Tabu Restrictions

A move is tabu if:

(R1) x_j changes from 1 to 0 (where x_j previously changed from 0 to 1).

(R2) x_k changes from 0 to 1 (where x_k previously changed from 1 to 0).

(R3) at least one of (R1) and (R2) occur. (This condition is more restrictive than either (R1) or (R2) separately—i.e. it makes more moves tabu.)

(R4) both (R1) and (R2) occur. (This condition is less restrictive than either (R1) or (R2) separately—i.e. it makes fewer moves tabu.)

(R5) both (R1) and (R2) occur, and in addition the reverse of these moves occurred simultaneously on the same iteration in the past. (This condition is less restrictive than (R4).)

(R6) $g(x)$ receives a value v' that it received on a previous iteration (i.e. $v' = g(x')$ for some previously visited solution x').

(R7) $g(x)$ changes from v'' to v', where $g(x)$ changed from v' to v'' on a previous iteration (i.e. $v' = g(x')$ and $v'' = g(x'')$ for some pair of solutions x' and x'' previously visited in sequence.)

Among the restrictions of these examples, only (R5) applies to a composite attribute, in which two component attributes simultaneously identify a single attribute of a previous move. (However, (R4) is meaningful only if the present move is composed of two such attributes, but does not depend on the condition that both of these attributes have occurred together in the past.) Also, while (R7) is less restrictive than (R6) (since it renders fewer moves tabu), both of these restrictions can reduce either to (R1) or (R2) by specifying $g(x) = x_j$ or $g(x) = x_k$. (Restriction (R6) is equivalent to (R7) in the situation where $g(x)$ can only take two different values.)

Tabu restrictions are also sometimes used to prevent repetitions rather than reversals, as illustrated by stipulating in (R1) that x_j previously changed from 1 to 0, rather than from 0 to 1. These have a rôle of preventing the repetition of a search path that leads away from a given solution. By contrast, restrictions that prevent reversals have a rôle of preventing a return to a previous solution. Hence, tabu restrictions vary according to whether they are defined in terms of reversals or duplications of their associated attributes.

The rôle of tabu status

A tabu restriction is typically activated only in the case where its attributes occurred within a limited number of iterations prior to the

present iteration (creating a recency-based restriction), or occurred with a certain frequency over a longer span of iterations (creating a frequency-based restriction). More precisely, a tabu restriction is enforced only when the attributes underlying its definition satisfy certain thresholds of recency or frequency. To exploit this notion, we define an attribute to be tabu-active when its associated reverse (or duplicate) attribute has occurred within a stipulated interval of recency or frequency in past moves. An attribute that is not tabu-active is called tabu-inactive.

The condition of being tabu-active or tabu-inactive is called the *tabu status* of an attribute. Sometimes an attribute is called tabu or not tabu to indicate that it is tabu-active or tabu-inactive. It is important to keep in mind in such cases that a 'tabu attribute' does not correspond to a tabu move. As the preceding examples show, a move may contain tabu-active attributes, but still may not be tabu if these attributes are not of the right number or kind to activate a tabu restriction.

The most common tabu restrictions, whose attributes are the reverse of those defining these restrictions, characteristically have a goal of preventing cycling and of inducing vigour into the search. However, some types of restrictions must be accompanied by others, at least periodically, to achieve the cycle avoidance effect. For example, the restriction (R5) is not able to prevent cycling by itself, regardless of the interval of time it is allowed to be in effect. This can be demonstrated by letting the ordered pair (j, k) denote an attribute in which x_j changes from 0 to 1 and x_k changes from 1 to 0. Then a sequence of 3 moves that creates the three attributes (1,2), (2,3), and (3,1) both starts and ends at the same solution, but this sequence is not prevented by restriction (R5). (R7) also may not prevent cycling, if $g(x)$ can change from a later value to an earlier value without visiting values that were successively generated at intermediate points (e.g. going from 5 to 10 to 15 and then back to 5, jumping over the reverse move from 15 to 10).

Cycle avoidance can easily be achieved over the duration of tabu tenure, however, by focusing specifically on *from-attributes* and *to-attributes* rather than on their ordered pair combinations. More precisely, as long as at least one *to-attribute* of a current move is not a *from-attribute* of a previous move, cycling cannot occur. Exami-

nation of the preceding restrictions shows that all except (R5) and (R7) implicitly are based on the requirement that specified *from-attributes* of previous moves must not be *to-attributes* of the current move, or else the move is tabu. (The only component attributes of the present move that are relevant to its tabu classification are its *to-attributes*, which to prevent reversals must be *from-attributes* of previous moves.)

It should be pointed out, however, that cycle avoidance is not an ultimate goal of the search process. In some instances, a good search path will result in revisiting a solution encountered before. The broader objective is to continue to stimulate the discovery of new high quality solutions. Hence in the longer term the issue of cycle avoidance is more subtle than simply preventing a solution from being revisited. The way that tabu restrictions depend on different choices of move attributes, and the consequences of this dependency, are examined in the following example.

An Example Consider a past move that involves a change from $x_j = p$ to $x_j = q$. To avoid a reversal, we stipulate that the *from-attribute* of this move, $x_j = p$, is tabu-active, thus allowing the possibility of preventing a move with a change in which $x_j = p$ is the *to-attribute*. But $x_j = p$ is not the only component of the past move that can qualify as a *from-attribute*, and hence that can be the basis for defining a tabu-active status.

By conceiving an attribute change implicitly to involve replacing an attribute e by a complementary attribute $\bar{e}$, the change from $x_j = p$ to $x_j = q$ in fact may be viewed as composed of two such attribute changes: from $x_j = p$ to $x_j \neq p$, and from $x_j \neq q$ to $x_j = q$. Thus, $x_j \neq q$ can also be regarded as a *from-attribute* of this change. By avoiding either of the tabu-active reverse attributes, to $x_j = p$ or to $x_j \neq q$, the present move will not be able to re-visit the solution that initiated the past move. (Note that avoiding $x_j \neq q$ is the same as compelling $x_j = q$, which is more restrictive than avoiding $x_j = p$.)

The problem illustrated at the start of this chapter gives an instructive example of options created by identifying tabu attributes in this way. The swap moves of the illustration consist of selecting two items, j and k, where items j and k occupy positions p and q respectively, and then exchanging their positions. Let $x_u = v$ denote

the statement 'item u is assigned to position v'. Hence the the swap move for interchanging the positions of items j and k can be represented as consisting of the two operations 'from $x_j = p$ to $x_j = q$' and 'from $x_k = q$ to $x_k = p$'. Subdividing these operations into their components, we can express the outcome as consisting of the following changes:

$$\begin{aligned}&\text{from } x_j = p \text{ to } x_j \neq p\\ &\text{from } x_j \neq q \text{ to } x_j = q\\ &\text{from } x_k = q \text{ to } x_k \neq q\\ &\text{from } x_k \neq p \text{ to } x_k = p.\end{aligned}$$

Thus, any combination of the preceding *from-attributes* can be selected to represent corresponding *to-attributes* of a move currently under consideration, for the purpose of defining a tabu restriction applicable to this move. We may elect, for instance, to rely on just the first and third of the preceding *from-attributes*, using the tabu restriction that classifies a move tabu only if it contains both $x_j = p$ and $x_k = q$ as *to-attributes*. (Hence this prevents the current move if it transfers item j to position p and item k to position q, where items j and k were respectively moved out of these two positions in the past, though not necessarily on the same move.) This is a weaker restriction than one based on either the second or fourth *from-attribute* above, which renders a move tabu if it contains $x_j \neq q$ or $x_k \neq p$ as a *to-attribute*, hence essentially compelling the current move to result in $x_j = q$ or $x_k = p$ (or possibly both, depending on the restriction chosen). One implication of choosing stronger or weaker tabu restrictions is to render smaller or larger tabu tenures appropriate.

Effect of Variable Codings Different codings of variables also lead to different consequences for creating tabu restrictions. For example, if $x_u = v$ instead is given the interpretation 'item u immediately precedes item v', then the swap of items j and k yields an altered set of attributes with different associated possibilities. Denoting the two items that immediately precede and immediately follow j by p and q, and the corresponding items for k by r and s, we see that the swap creates the following changes:

$$\text{from } x_j = q \text{ to } x_j = s$$

$$\text{from } x_k = s \text{ to } x_k = q$$
$$\text{from } x_p = j \text{ to } x_p = k$$
$$\text{from } x_r = k \text{ to } x_r = j.$$

Moreover, each of these subdivides into two additional components (for example, the first becomes 'from $x_j = q$ to $x_j \neq q$' and 'from $x_j \neq s$ to $x_j = s$'), yielding a set of options for defining tabu restrictions that is considerably expanded over those of the preceding coding of the variables.

Representationally, there may be multiple options for characterizing the same set of attributes, and it is appropriate to use one that is natural for the problem setting. In this case, for example, it is convenient to represent the condition 'item u immediately precedes item v' as an arc (u, v) from node u to node v in a directed graph, and by this convention the statement 'from $x_j = q$ to $x_j = s$' corresponds to saying 'arc (j, q) replaces arc (j, s)'. A component change of the form 'from $x_j = q$ to $x_j \neq q$' (or 'from $x_j \neq q$ to $x_j = q$') then corresponds to saying that arc (j, q) is dropped from (or added to) the graph. We note it is always possible to encode the pair of conditions $x_j = q$ and $x_j \neq q$ as the assignment of values to a binary variable, e.g. letting $x_{jq} = 1$ denote $x_j = q$ and letting $x_{jq} = 0$ denote $x_j \neq q$, and in the present example this yields the standard algebraic notation for expressing that arc (j, q) is absent or present in a graph.

Broadly speaking, regardless of the representation employed, a move can be determined to be tabu by a restriction defined over any set of conditions on its attributes, provided these attributes are currently tabu-active. As the preceding discussion illustrates, a common type of restriction operates by selecting some subset of attributes and declaring the move to be tabu if a certain minimum number (e.g. one or all) are tabu-active.

3.2.6 Recency-based tabu memory functions

To keep track of the status of move attributes that compose tabu restrictions, and to determine when these restrictions are applicable, several basic kinds of memory functions have been found useful. Two common examples of recency-based memory functions are specified by the arrays *tabu_start*(e) and *tabu_end*(e), where e ranges over attributes relevant to a particular application. These arrays respec-

tively identify the starting and ending iterations of the tabu tenure for attribute e, thus bracketing the period during which e is tabu-active.

The rule for identifying appropriate values for $tabu\_start(e)$ and $tabu\_end(e)$ results from keeping track of the attributes at each iteration that are components of the current move. In particular, on iteration i, if e is an attribute of the current move, and tabu status is defined to avoid reversals, then we set $tabu\_start(\overline{e}) = i + 1$, indicating that the reverse attribute begins its tabu-active status at the start of the next iteration. (For example, if e represents 'from $x_j = p$' then $\overline{e}$ can represent 'to $x_j = p$'.) Attribute $\overline{e}$ will retain this status throughout its tabu tenure, which we denote by t. This then yields $tabu\_end(\overline{e}) = i + t$, so that the tenure for $\overline{e}$ ranges over the t iterations from $i + 1$ to $i + t$.

As a result, it is easy to test whether an arbitrary attribute e is tabu-active, by checking to see if $tabu\_end(e) \geq current\_iteration$. By initializing $tabu\_end(e) = 0$ for all attributes, we insure that $tabu\_end(e) < current\_iteration$, and hence that attribute e is tabu-inactive, until the update previously specified is performed. This suggests we need to keep only the single array $tabu\_end(e)$ to provide information about tabu status. However, we will see that situations arise where it is valuable to keep $tabu\_start(e)$, and either to infer $tabu\_end(e)$ by adding an appropriate value of t (currently computed, or preferably extracted from a pre-stored sequence), or to maintain $tabu\_end(e)$ as a separate array.

Memory can often be further simplified when attributes represent binary alternatives, such as changing from $x_j = 0$ to $x_j = 1$. Then, instead of recording a separate value $tabu\_start(e)$ for each of these attributes, it suffices simply to record a single value $tabu\_start(j)$. We automatically know whether $tabu\_start(j)$ refers to changing from $x_j = 0$ to $x_j = 1$ or the reverse, by taking account of the value of x_j in the current solution. If currently $x_j = 1$, for example, the most recent change was from $x_j = 0$ to $x_j = 1$. Then the reverse attribute, derived from changing x_j from 1 to 0, is the one whose tenure is represented by the value of $tabu\_start(j)$. (We assume that the latest tabu tenure assigned to an attribute takes precedence over all others.)

Regardless of the data structure employed, the key issue for creating tabu status using recency-based memory is to determine a 'good

value' of t. Rules for determining t are classified as static or dynamic. *Static rules* choose a value for t that remains fixed throughout the search. *Dynamic rules* allow the value of t to vary. Examples of these two kinds of rules are as follows.

Illustrative Rules to Create Tabu Tenure
(Recency Based)

Static rules	Choose t to be a constant such as $t = 7$ or $t = \sqrt{n}$, where n is a measure of problem dimension.
Dynamic rules	*Simple dynamic*: Choose t to vary (randomly or by systematic pattern) between bounds t_{min} and t_{max}, such as $t_{min} = 5$ and $t_{max} = 11$ or $t_{min} = .9\sqrt{n}$ and $t_{max} = 1.1\sqrt{n}$.
	Attribute-dependent dynamic: Choose t as in the Simple dynamic rule, but determine t_{min} and t_{max} to be larger for attributes that are more attractive, e.g. based on quality or influence considerations.

The indicated values such as 7 and $\sqrt{n}$ are only suggestive, and represent parameters whose preferred values should be set by experimentation for a particular class of problems. Values between 7 and 20 in fact appear to work well for a variety of problem classes, while values between $.5\sqrt{n}$ and $2\sqrt{n}$ appear to work well for other classes. (A weighted multiple of $\sqrt{n}$ is replaced by a weighted multiple of n for some problems.) As previously intimated, if $tabu\_end(e)$ is not maintained separately, but is inferred as the value $tabu\_start(e) + t$, then for the dynamic case it may be preferable to pre-compute a sequence of appropriate values for t and simply step through them each time a new t is needed. (Random sequences can be reasonably approximated this way with considerable saving of computational effort. Alternatively, t can be computed only once or a small number of times on a given iteration, instead of being recomputed separately for each trial move.)

It is often appropriate to allow different types of attributes defining a tabu restriction to be given different values for the tenure t. For example, some attributes can contribute more strongly to a tabu restriction than others, and should be given a briefer tabu tenure to avoid making the restriction too severe. To illustrate, consider a

problem of identifying an optimal subset of m items from a much larger set of n items. (For instance, such a problem may involve identifying a subset of m edges from an n-edge graph to create a travelling salesman tour, or a subset of m locations from n available sites to establish distribution centres, or a subset of m nodes from an n-node complex to serve as telecommunication switching centres, etc.) Suppose each move consists of exchanging one or a small number of items in the subset with an equal number outside the subset, to create a new subset of m items. Accompanying this, also suppose a tabu restriction is used that forbids a move if it contains either an item recently added or an item recently dropped, where the tabu tenure provides the meaning of 'recently'.

If the tenure for added and dropped items is the same, the preceding restriction can become very lopsided. In particular, when other factors are equal, preventing items in the subset from being dropped is much more restrictive than preventing items not in the subset from being added, since there are far fewer contained in the subset than contained outside. In addition, preventing elements added to the subset from being dropped for a relatively long time can significantly inhibit available choices; hence the tenure for these elements should be made small by comparison to the tenure for preventing elements dropped from the subset from being added, whether by using static or dynamic rules.

Practical experience indicates that dynamic rules are typically more robust than static rules (see, e.g. Glover *et al.* [5]). Good parameter values for dynamic rules normally range over a wider interval, and produce results comparable or superior to the outcomes produced by static rules. Dynamic rules that depend on both attribute type and quality, where greater tenures are allotted to prevent reversals of attributes that participate in high quality moves, have proved quite effective for difficult problems related to scheduling and routing (Dell'Amico and Trubian [6]; Gendreau *et al.* [7]; Laguna *et al.* [25]). In addition, a class of dynamic rules based on introducing moving gaps in tenure, and another class based on exploiting logical relationships underlying attribute sequences, have recently proved effective (Chakrapani and Skorin-Kapov [9]; Dammeyer and Voss [10]). Dynamic rules for determining tabu tenure are among the aspects of tabu search that deserve more study.

3.2.7 Aspiration criteria

Aspiration criteria are introduced in tabu search to determine when tabu restrictions can be overridden, thus removing a tabu classification otherwise applied to a move. The appropriate use of such criteria can be very important for enabling a TS method to achieve its best performance levels.

Early applications employed only a simple type of aspiration criterion, consisting of removing a tabu classification from a trial move when the move yields a solution better than the best obtained so far. (Such a rule is illustrated in the example of Section 3.2.1.) This criterion remains widely used. However, other aspiration criteria can also prove effective for improving the search.

A basis for one of these criteria arises by introducing the concept of *influence*, which measures the degree of change induced in solution structure or feasibility. (Influence is often associated with the idea of move distance, i.e. a move of greater distance is conceived of as having greater influence—see [5].) This notion can be illustrated for a problem of distributing unequally weighted objects among boxes, where the goal is to give each box as nearly as possible the same weight. A high influence move, which significantly changes the structure of the current solution, is exemplified by a move that transfers a heavy weight object from one box to another, or that swaps objects of very dissimilar weights between two boxes. Such a move may or may not improve the current solution, though it is less likely to yield an improvement when the current solution is relatively good. But high influence moves are important, especially during intervals of breaking away from local optimality, because a series of moves that is confined to making only small structural changes is unlikely to uncover a chance for significant improvement. (Such an effect is illustrated in job sequencing problems by exchanging positions of jobs that are close together.)

Moves of lower influence may normally be tolerated until the opportunities for gain from them appear to be negligible. At such a point, and in the absence of improving moves, aspiration criteria should shift to give influential moves a higher rank. Also, once an influential move is made, tabu restrictions previously established for less influential moves should be dropped or 'weakened', in a manner to be explained. (Bias that may be employed to favour the choice of

other influential moves should likewise be temporarily diminished.) These considerations of move influence interact with considerations of regionality and search direction, as indicated below.

Aspirations are of two kinds: *move aspirations* and *attribute aspirations.* A move aspiration, when satisfied, revokes the move's tabu classification. An attribute aspiration, when satisfied, revokes the attribute's tabu-active status. In the latter case the move may or may not change its tabu classification, depending on whether the tabu restriction can be activated by more than one attribute.

The table below lists criteria for determining the admissibility of a trial solution, x^{trial}, as a candidate for consideration (potentially to become x^{next}), where x^{trial} is generated by a move that ordinarily would be classified tabu. The first of these criteria is rarely applicable, but is understood automatically to be part of any tabu search procedure. These aspiration criteria include several useful strategies for tabu search that have not yet been widely examined and that warrant fuller investigation.

For example, a special case of the Regional Aspiration by Objective occurs by defining $R = \{x : g(x) = r\}$, where $g(x)$ is a hashing function created to distinguish among different x vectors according to the value assigned to $g(x)$. (E.g. $g(x)$ can be an integer-valued function defined modulo p, taking the values $r = 0, 1, \ldots, p-1$.) Then $best\_cost(R)$ is conveniently recorded as $best\_cost(r)$, identifying the minimum $c(x)$ found when $g(x) = r$. The 'regionality' defined by R in this case provides a basis for integrating the elements of aspiration and differentiation. (A $g(x)$ hashing function can also be treated as an attribute function, and incorporated into tabu restrictions as described earlier. Or in reverse, a hashing function can be defined over attributes, with particular emphasis on those that qualify as influential.) Such an approach can be employed to complement uses of hashing functions in tabu search suggested by Hansen and Jaumard [11] and by Woodruff and Zemel [12].

Aspiration by Search Direction and Aspiration by Influence provide attribute aspirations rather than move aspirations. In most cases attribute and move aspirations are equivalent. (Among the tabu restrictions (R1) to (R7) of Section 3.2.5, only (R3) can provide conditions where these two types of aspirations differ, i.e. where an attribute may be tabu-inactive without necessarily revoking the tabu

classification of the associated move.) Nevertheless, different means are employed for testing these two kinds of aspirations.

Illustrative Aspiration Criteria

Aspiration by Default: If all available moves are classified tabu, and are not rendered admissible by some other aspiration criteria, then a 'least tabu' move is selected. (For example, select a move that loses its tabu classification by the least increase in the value of *current_iteration*, or by an approximation to this condition.)

Aspiration by Objective: Global form (customarily used): A move aspiration is satisfied, permitting x^{trial} to be a candidate for selection, if $c(x^{trial}) < best\_cost$.
Regional form: Subdivide the search space into regions $R \in \mathbf{R}$, identified by bounds on values of functions $g(x)$ (or by time intervals of search). Let $best\_cost(R)$ denote the minimum $c(x)$ for x found in R. Then, for $x^{trial} \in R$, a move aspiration is satisfied (for moving to x^{trial}) if $c(x^{trial}) < best\_cost(R)$.

Aspiration by Search Direction: Let $direction(e) = improving$ if the most recent move containing $\overline{e}$ was an improving move, and $direction(e) = nonimproving$, otherwise. ($direction(e)$ and $tabu\_end(e)$ are set to their current values on the same iteration.) An attribute aspiration for e is satisfied (making e tabu-inactive) if $direction(e) = improving$ and the current trial move is an improving move, i.e. if $c(x^{trial}) < c(x^{now})$.

Aspiration by Influence: Let $influence(e) = 0$ or 1 according to whether the move that establishes the value of $tabu\_start(e)$ is a low influence move or a high influence move ($influence(e)$ is set at the same time as setting $tabu\_start(e)$). Also, let $latest(L)$, for $L = 0$ or 1, equal the most recent iteration that a move of influence level L was made. Then an attribute aspiration for e is satisfied if $influence(e) = 0$ and $tabu\_start(e) < latest(1)$. ($e$ is associated with a low influence move, and a high influence move has been performed since establishing the tabu status for e.) For multiple influence levels, $L = 0, 1, 2, \ldots$, the aspiration for e is satisfied if there is an $L > influence(e)$ such that $tabu\_start(e) < latest(L)$.

Aspiration criteria refinements

Refinements of the criteria illustrated above provide an opportunity to enhance the power of tabu search for applications that are more complex, or that offer a large reward for solutions of very high quality. We identify some of the possibilities for achieving this in the following.

Creating a tabu status that varies by degrees, rather than simply designating an attribute to be tabu-active or tabu-inactive, leads to an additional refinement of Aspiration by Search Direction and Aspiration by Influence. Graduated tabu status is implicit in the penalty function and probabilistic variants of tabu search, where status is customarily expressed as a function of how recently or frequently an attribute has become tabu-active. However, to employ this idea to enhance the preceding aspiration criteria, we create a single additional intermediate tabu state that falls between the two states of tabu-active and tabu-inactive. In particular, when an aspiration is satisfied for an attribute that otherwise is tabu-active, we call it a *pending tabu attribute.*

A move that would be classified tabu if its pending tabu attributes are treated as tabu-active, but that would not be classified tabu otherwise, is correspondingly called a *pending tabu move.* A pending tabu move can be treated in one of two ways. In the least restrictive approach, such a move is not prevented from being selected, but is shifted in status so that it will only be a candidate for selection if no improving moves exist except those that are tabu. In the more moderate approach, a pending tabu move additionally must be an improving move to qualify for selection. (This will occur automatically for Aspiration by Search Direction, since in this case a move can only become a pending tabu move when it is improving.)

An Aspiration consequence for Strong Admissibility The preceding notions lead to an additional type of aspiration. Define a move to be strongly admissible if:

(1) it is admissible to be selected and does not rely on aspiration criteria to qualify for admissibility; or

(2) it qualifies for admissibility based on the Global Aspiration by Objective, by satisfying $c(x^{trial}) < best\_cost$.

Aspiration by Strong Admissibility: Let *last_nonimprovement* equal the most recent iteration that a nonimproving move was made, and let *last_strongly_admissible* equal the most recent iteration that a strongly admissible move was made. Then, if *last_nonimprovement* < *last_strongly_admissible*, reclassify every improving tabu move as a pending tabu move (thus allowing it to be a candidate for selection if no other improving moves exist).

The inequality *last_nonimprovement* < *last_strongly_admissible* of the preceding aspiration condition implies two things: first that a strongly admissible improving move has been made since the last nonimproving move, and second that the search is currently generating an improving sequence. (The latter results since only improving moves can occur on iterations more recent than *last_nonimprovement*, and the set of such iterations is nonempty.)

This type of aspiration ensures that the method will always proceed to a local optimum whenever an improving sequence is created that contains at least one strongly admissible move. In fact, condition (2) defining a strongly admissible move can be removed without altering this effect, since once the criterion $c(x^{trial}) < best\_cost$ is used to justify a move selection, then it will continue to be satisfied by all improving moves on subsequent iterations until a local optimum is reached.

Because of its extended ability to override tabu status, the Aspiration by Strong Admissibility may be predicated on the requirement that a move with a high influence level has been made since the end of the most recent (previous) improving sequence. Specifically, such a high influence move should have occurred on an iteration greater than the most recent iteration prior to *last_nonimprovement* on which an improving move was executed. This added requirement is applicable whether or not Aspiration by Influence is used.

These ideas can be used to generate an *alternating TS method* related to the tabu thresholding approach of Glover [13]. Such a method results by adding a further condition to the Aspiration by Strong Admissibility, stipulating that once a nonimproving move is executed, then no improving move is allowed unless it is strongly admissible, thereby generating what may be called an alternating tabu

path. The consequence is that each improving sequence in such an alternating tabu path terminates with a local optimum. (An Aspiration by Default must also be considered a strongly admissible move to assure this in exceptional cases.)

The effect of tabu status in this alternating approach can be amplified during a nonimproving phase by interpreting the value *tabu_end*(e) to be shifted to a larger value for all attributes e, until a strongly admissible move is executed and the phase ends. Recent results by Ryan [14] on the depth and width of paths linking local optima are relevant to determining ranges for shifting *tabu_end*(e) in such alternating constructions.

Special considerations for Aspiration by Influence

The Aspiration by Influence criterion can be modified to create a considerable impact on its effectiveness for certain types of applications. The statement of this aspiration derives from the observation that a move characteristically is influential by virtue of containing one or more influential attributes (jobs with large set-up or processing times, warehouses with large capacities, circuits with multiple switches, etc.). Under such conditions, it is appropriate to consider levels of influence defined over attributes, as expressed by *influence(e)*. In other cases, however, a move may derive its influence from the unique combination of attributes involved, and Aspiration by Influence then preferably translates into a move aspiration rather than an attribute aspiration. (In some instances the attribute orientation can be maintained by defining *influence(e)* to be the influence of the trial move that contains e.)

More significantly, in many applications influence depends on a form of connectivity, causing its effects to be expressed primarily over a particular range. We call this range the *sphere of influence* of the associated move or attribute. For example, in the problem of distributing weighted objects among boxes, a move that swaps objects between two boxes has a relatively narrow sphere of influence, affecting only those future moves that transfer an object into, or out of, one of these two boxes. Accordingly, under such circumstances Aspiration by Influence should be confined to modifying the tabu status of attributes, or the tabu classification of moves, that fall within an associated sphere of influence. In the example of swapping objects

between boxes, the attributes rendered tabu-inactive would be restricted to *from-attributes* associated with moving an object out of one of the two boxes and *to-attributes* associated with moving an object into one of these boxes. The change of tabu status continues to depend on the conditions noted previously. The influence of the attribute (or move containing it) must be less than that of the earlier move, and the iteration $tabu\_start(e)$ for the attribute must precede the iteration on which the earlier influential move occurred. These conditions can be registered by setting a flag for $tabu\_start(e)$ when the influential move is executed, without having to check again later to see if e is affected by such a move. When $tabu\_start(e)$ becomes reassigned a new value, the flag is dropped.

As the preceding observations suggest, effective measures of move influence and associated characterizations of spheres of influence are extremely important. In addition, it should be noted that influence can be expressed as a function of tabu search memory components, as where a move containing attributes that have neither recently nor frequently been tabu-active may be classified as more highly influential (because executing the move will change the tabu status of these attributes more radically). This encourages a dynamic definition of influence, which varies according to the current search state. These multiple aspects of move influence are likely to constitute a more significant area for future investigation in tabu search.

3.2.8 Frequency-based memory

Frequency-based memory provides a type of information that complements the information provided by recency-based memory, broadening the foundation for selecting preferred moves. Like recency, frequency is often weighted or decomposed into subclasses by taking account of the dimensions of solution quality and move influence.

For our present purposes, we conceive frequency measures to consist of ratios, whose numerators represent counts of the number of occurrences of a particular event (e.g. the number of times a particular attribute belongs to a solution or move) and whose denominators generally represent one of four types of quantities, as shown below.

Denominators for Frequency Measures

(D1)	The total number of occurrences of all events represented by the numerators (such as the total number of associated iterations).
(D2)	The sum of the numerators.
(D3)	The maximum numerator value.
(D4)	The average numerator value.

Denominators (D3) and (D4) give rise to what may be called relative frequencies. The meaning of these different types of frequencies will be clarified by examples below. In cases where the numerators represent weighted counts, some of which may be negative, (D3) and (D4) are expressed as absolute values and (D2) is expressed as a sum of absolute values (possibly shifted by a small constant to avoid a zero denominator).

Let $x(1), x(2), \ldots, x(\mathit{current\_iteration})$ denote the sequence of solutions generated to the present point of the search process, and let S denote a subsequence of this solution sequence. We take the liberty of treating S as a set as well as an ordered sequence. Elements of S are not necessarily consecutive elements of the full solution sequence. (For example, we sometimes will be interested in cases where S consists of different subsets of high quality local optima.)

Notationally, we let $S(x_j = p)$ denote the set of solutions in S for which $x_j = p$, and let $\#S(x_j = p)$ denote the cardinality of this set (hence the number of times x_j receives the value p over $x \in S$). Similarly, let $S(x_j = p \text{ to } x_j = q)$ denote the set of solutions in S that result by a move that changes $x_j = p$ to $x_j = q$. Finally, let $S(\text{from } x_j = p)$ and $S(\text{to } x_j = q)$ denote the sets of solutions in S that respectively contain $x_j = p$ as a *from-attribute* or $x_j = q$ as a *to-attribute* (for a move to the next solution, or from the preceding solution, in the sequence $x(1), \ldots, x(\mathit{current\_iteration})$). In general, if *solution_attribute* represents any attribute of a solution that can take the rôle of a *from-attribute* or a *to-attribute* for a move, and if *move_attribute* represents an arbitrary move attribute denoted by (*from-attribute, to-attribute*), then

$S(solution\_attribute)$ = $\{x \in S : x$ contains *solution_attribute* $\}$
$S(move\_attribute)$ = $\{x \in S : x$ results from a move containing *move_attribute* $\}$
$S(from\text{-}attribute)$ = $\{x \in S : x$ initiates a move containing *from-attribute* $\}$
$S(to\text{-}attribute)$ = $\{x \in S : x$ results from a move containing *to-attribute* $\}$.

The quantity $\#S(x_j = p)$ constitutes a *residence measure*, since it identifies the number of times the attribute $x_j = p$ resides in the solutions of S. Correspondingly, we call a frequency that results by dividing such a measure by one of the denominators (1) to (4) a *residence frequency*. For the numerator $\#S(x_j = p)$, the denominators (1) and (2) both correspond to $\#S$, while denominators (3) and (4) respectively are given by Max $(\#S(x_k = q) : \forall k, q)$ and by Mean $(\#S(x_k = q) : \forall k, q)$.

The quantities $\#S(x_j = p$ to $x_j = q)$, $\#S($from $x_j = p)$ and $\#S($to $x_j = q)$ constitute *transition measures*, since they identify the number of times x_j changes from and/or to specified values. Likewise, frequencies based on such measures are called *transition frequencies*. Denominators for creating such frequencies from the foregoing measures include $\#S$, the total number of times the indicated changes occur over S for different j, p and/or q values, and associated Max and Mean quantities.

Distinctions between frequency types

Residence frequencies and transition frequencies sometimes convey related information, but in general carry different implications. They are sometimes confused (or treated identically) in the literature. A noteworthy distinction is that residence measures, by contrast to transition measures, are not concerned with whether a particular solution attribute of an element $x(i)$ in the sequence S is a *from-attribute* or a *to-attribute*, or even whether it is an attribute that changes in moving from $x(i)$ to $x(i+1)$ or from $x(i-1)$ to $x(i)$. It is only relevant that the attribute can be a *from-attribute* or a *to-attribute* in some future move. Such measures can yield different types of implications depending on the choice of the subsequence S.

A high residence frequency, for example, may indicate that an attribute is highly attractive if S is a subsequence of high quality

solutions, or may indicate the opposite if S is a subsequence of low quality solutions. On the other hand, a residence frequency that is high (or low) when S contains both high and low quality solutions may point to an entrenched (or excluded) attribute that causes the search space to be restricted, and that needs to be jettisoned (or incorporated) to allow increased diversity.

From a computational standpoint, when S consists of all solutions generated after a specified iteration, then a residence measure can be currently maintained and updated by reference to values of the *tabu_start* array, without the need to increment a set of counters at each iteration. For a set S whose solutions do not come from sequential iterations, however, residence measures are calculated simply by running a tally over elements of S.

Transition measures are generally quite easy to maintain by performing updates during the process of generating solutions (assuming the conditions defining S, and the attributes whose transition measures are sought, are specified in advance). This results from the fact that typically only a few types of attribute changes are considered relevant to track when one solution is replaced by the next, and these can readily be isolated and recorded. The frequencies in the example of Section 3.2.1 constitute an instance of transition frequencies that were maintained in this simple manner. Their use in this example, however, encouraged diversity by approximating the type of rôle that residence frequencies are usually better suited to take.

As a final distinction, a high transition frequency, in contrast to a high residence frequency, may indicate an associated attribute is a 'crack filler', that shifts in and out of solution to perform a fine tuning function. Such an attribute may be interpreted as the opposite of an influential attribute, as considered earlier in the discussion of Aspiration by Influence. In this context, a transition frequency may be interpreted as a measure of volatility.

Examples and uses of frequency measures

Illustrations of both residence and transition frequencies are as follows. (Only numerators are indicated, understanding denominators to be provided by conditions (1) to (4) above.)

Example Frequency Measures (Numerators)

(F1)	$\#S(x_j = p)$
(F2)	$\#S(x_j = p$ for some $x_j)$
(F3)	$\#S($to $x_j = p)$
(F4)	$\#S(x_j$ changes), i.e. $\#S($from-or-to $x_j = p$ for some $p)$
(F5)	$\sum_{x \in S(x_j = p)} c(x)/\#S(x_j = p)$
(F6)	Replace $S(x_j = p)$ in (F5) with $S(x_j \neq p$ to $x_j = p)$
(F7)	Replace $c(x)$ in (F6) with a measure of the influence of the solution attribute $x_j = p$

The measures (F5) - (F7) implicitly are weighted measures, created by reference to solution quality in (F5) and (F6), and by reference to move influence in (F7) (or more precisely, influence of an attribute composing a move). Measure (F5) may be interpreted as the average $c(x)$ value over S when $x_j = p$. This quantity can be directly compared to other such averages or can be translated into a frequency measure using denominators such as the sum or maximum of these averages.

Attributes that have greater frequency measures, just as those that have greater recency measures (i.e. that occur in solutions or moves closer to the present), can initiate a tabu-active status if S consists of consecutive solutions that end with the current solution. However, frequency-based memory typically finds its most productive use as part of a longer term strategy, which employs incentives as well as restrictions to determine which moves are selected. In such a strategy, restrictions are translated into evaluation penalties, and incentives become evaluation enhancements, to alter the basis for qualifying moves as attractive or unattractive.

To illustrate, an attribute such as $x_j = p$ with a high residence frequency may be assigned a strong incentive ('profit') to serve as a *from-attribute*, thus resulting in the choice of a move that yields $x_j \neq p$. Such an incentive is particularly relevant in the case where $tabu\_start(x_j \neq p)$ is small, since this value identifies the latest iteration that $x_j \neq p$ served as a *from-attribute* (for avoiding reversals), and hence discloses that $x_j = p$ has been an attribute of every solution since.

Frequency-based memory therefore is usually applied by introducing graduated tabu states, as a foundation for defining penalty and incentive values to modify the evaluation of moves. A natural

connection exists between this approach and the recency-based memory approach that creates tabu status as an all-or-none condition. If the tenure of an attribute in recency-based memory is conceived of as a conditional threshold for applying a very large penalty, then the tabu classifications produced by such memory can be interpreted as the result of an evaluation that becomes strongly inferior when the penalties are activated. It is reasonable to anticipate that conditional thresholds should also be relevant to determining the values of penalties and incentives in longer term strategies. Most applications at present, however, use a simple linear multiple of a frequency measure to create a penalty or incentive term. Fundamental ways for taking advantage of frequency based memory are indicated in the next section.

3.2.9 Frequency-based memory in simple intensification and diversification processes

The rôles of intensification and diversification in tabu search are already implicit in several of the preceding prescriptions, but they become especially relevant in longer term search processes. Intensification strategies undertake to create solutions by aggressively encouraging the incorporation of 'good attributes'. In the short term this consists of incorporating attributes receiving highest evaluations by the approaches and criteria previously described, while in the intermediate to long term it consists of incorporating attributes of solutions from selected élite subsets (implicitly focusing the search in subregions defined relative to these subsets). Diversification strategies instead seek to generate solutions that embody compositions of attributes significantly different from those encountered previously during the search. These two types of strategies counterbalance and reinforce each other in several ways.

We first examine simple forms of intensification and diversification approaches that make use of frequency-based memory. These approaches will be illustrated by reference to residence frequency measures, but similar observations apply to the use of transition measures, taking account of contrasting features previously noted.

For a diversification strategy we choose S to be a significant subset of the full solution sequence; for example, the entire sequence starting with the first local optimum, or the subsequence consisting of all

local optima. (For certain strategies based on transition measures, S may usefully consist of the subsequence containing each maximum unbroken succession of non-improving moves that immediately follow a local optimum, focusing on $S(to\_attribute)$ for these moves.)

For an intensification strategy we choose S to be a small subset of élite solutions (high quality local optima) that share a large number of common attributes, and secondarily whose members can reach each other by relatively small numbers of moves, independent of whether these solutions lie close to each other in the solution sequence. For example, collections of such subsets S may be generated by clustering procedures, followed by employing a parallel processing approach to treat each selected S separately.

Below we provide rules for generating a penalty or incentive function, PI, which apply equally to intensification and diversification strategies. However, the function PI creates a penalty for one strategy (intensification or diversification) if and only if it creates an incentive for the other. For illustrative purposes, suppose that a move currently under consideration includes two move attributes, denoted e and f, which further may be expressed as $e = (e\_from, e\_to)$ and $f = (f\_from, f\_to)$. To describe the function PI, we let $F(e\_from)$ and $F(e\_to)$ etc. denote the frequency measure for the indicated *from-attributes* and *to-attributes*, and let $T_1, T_2, \ldots, T_6$ denote selected positive thresholds, whose values depend on the case considered.

These conditions for defining PI are related to those previously illustrated to identify conditions in which attributes become tabu-active. For example, specifying that (1) must be positive to make PI positive corresponds to introducing a tabu penalty (or incentive) when both measures exceed their common threshold. If a measure is expressed as the duration since an attribute was most recently made tabu-active, and if the threshold represents a common limit for tabu tenure, then (1) can express a recency-based restriction for determining a tabu classification. Assigning different thresholds to different attributes in (1) corresponds to establishing attribute-dependent tabu tenures. Similarly, the remaining values (2) through (6) may be interpreted as analogues of values that define recency-based measures for establishing a tabu classification, implemented in this case by a penalty.

Illustrative Penalty/Incentive Function PI for To-attributes

Choose PI as a monotonic nondecreasing function of one of the following quantities, where PI is positive when the quantity is positive, and is 0 otherwise. (PI yields a penalty in a diversification strategy and an incentive in an intensification strategy.)

(1) $\text{Min}\{F(e\_to), F(f\_to)\} - T_1$

(2) $\text{Max}\{F(e\_to), F(f\_to)\} - T_2$

(3) $\text{Mean}\{F(e\_to), F(f\_to)\} - T_3$

Illustrative Penalty/Incentive Function PI for From-attributes

Choose PI as a monotonic nondecreasing function of one of the following quantities, where PI is positive when the quantity is positive, and is 0 otherwise. (PI yields an incentive in a diversification strategy and a penalty in an intensification strategy.)

(4) $\text{Min}\{F(e\_from), F(f\_from)\} - T_4$

(5) $\text{Max}\{F(e\_from), F(f\_from)\} - T_5$

(6) $\text{Mean}\{F(e\_from), F(f\_from)\} - T_6$

From these observations, it is clear that the frequency measure F may be extended to represent combined measures of both recency and frequency. Although these measures are already implicitly combined—when penalties and incentives based on frequency measures are joined with tabu classifications based on recency measures, as a foundation for selecting current moves—it is possible that other forms of combination are superior. For example, human problem-solving appears to rely on combinations of these types of memory that incorporate a time-discounted measure of frequency. Such considerations may lead to the design of more intelligent functions for capturing preferred combinations of these memory types.

3.3 Broader Aspects of Intensification and Diversification

Intensification and diversification approaches that use penalties and incentives represent only one class of such strategies. A larger collec-

tion emerges by direct consideration of intensification and diversification goals. We examine several approaches that have been demonstrated to be useful in previous applications, and also indicate approaches we judge to have promise in applications of the future. To begin, we make an important distinction between diversification and randomization.

3.3.1 Diversification versus randomization

Seeking a diversified collection of solutions is very different from seeking a randomized collection of solutions. In general, we are interested not just in diversified collections but also in diversified *sequences*, since often the order of examining elements is important in tabu search. This can apply, for example, where we seek to identify a sequence of new solutions (not seen before) so that each successive solution is *maximally diverse* relative to all solutions previously generated. This includes possible reference to a baseline set of solutions, such as $x \in S$, which takes priority in establishing the diversification objective (i.e. where the first level goal is to establish diversification relative to S, and then in turn relative to other solutions generated). The diversification concept applies as well to generating a diverse sequence of numbers or a diverse set of points from the vertices of a unit hypercube.

Let $Z(k) = \{z(1), z(2), \ldots, z(k)\}$ represent a sequence of points drawn from a set Z. For example, Z may be a line interval if the points are scalars. We take $z(1)$ to be a seed point of the sequence. (The seed point may be excepted from the requirement of belonging to Z.) Then we define $Z(k)$ to be a *diversified sequence* (or simply a *diverse sequence*), relative to a chosen distance metric d over Z by requiring each subsequence $Z(h)$ of $Z(k), h < k$, and each associated point $z = z(h+1)$ to satisfy the following hierarchy of conditions:

(A) z maximizes the minimum distance $d(z, z(i))$ for $i \leq h$;

(B) subject to (A), z maximizes the minimum distance $d(z, z(i))$ for $1 < i \leq h$, then for $2 < i \leq h, \ldots$, etc. (in strict priority order);

(C) subject to (A) and (B), z maximizes the distance $d(z, z(i))$ for $i = h$, then for $i = h-1, \ldots$, and finally for $i = 1$. (Additional ties may be broken arbitrarily.)

To handle diversification relative to an initial baseline set Z^* (such as a set of solutions $x \in S$), the preceding hierarchy of conditions is preceded by a condition stipulating that z first maximizes the minimum distance $d(z, z^*)$ for $z^* \in Z^*$. A useful (weaker) variant of this condition simply treats points of Z^* as if they constitute the last elements of the sequence $Z(h)$.

Variations on (A), (B), and (C), including going deeper in the hierarchy before arbitrary tie breaking, are evidently possible. Such conditions make it clear that a diverse sequence is considerably different from a random sequence. Further, they are computationally very demanding to satisfy. Even by omitting condition (B), and retaining only (A) and (C), if the elements $z(i)$ refer to points on a unit hypercube, then by our present state of knowledge the only way to generate a diverse sequence of more than a few points is to perform comparative enumeration. (However, a diverse sequence of points on a line interval, particularly if $z(1)$ is an endpoint or midpoint of the interval, can be generated with much less difficulty.) Because of this, it can sometimes be useful to generate sequences by approximating the foregoing conditions (see Glover [15]). Taking a broader view, an extensive effort to generate diverse sequences can be performed in advance, independent of problem solving efforts, so that such sequences are pre-computed and available as needed. Further, a diverse sequence for elements of a high dimensional unit hypercube may be derived by reverse projection techniques ('lifting' operations) from a sequence for a lower dimensional hypercube, ultimately making reference to sequences from a line interval.

Biased diversification, just as biased random sampling, is possible by judicious choices of the set Z. Also, while the goals of diversification and randomization are somewhat different, the computational considerations share a feature in common. To generate a random sequence by the strict definition of randomness would require massive effort. Years of study have produced schemes for generating sequences that empirically approximate this goal, and perhaps a similar outcome may be possible for generating diversified sequences. The hypothesis of tabu search, in any case, is that recourse to diversification is more appropriate (and more powerful) in the problem solving context than recourse to randomization.

We note these observations can be applied in a setting, as subse-

quently discussed, where the device of producing a solution 'distant from' another is accomplished not by reference to a standard distance metric, but rather by a series of displacements which involve selecting a move from a current neighbourhood at each step. (In this case the metric may derive from differences in weighted measures defined over *from-attributes* and *to-attributes.*) An application of these ideas is given in Kelly *et al.* [16], and we also discuss a special variation under the heading of 'Path Relinking' below. This stepwise displacement approach is highly relevant to those situations where neighbourhood structures are essential for preserving desired properties (such as feasibility).

3.3.2 Reinforcement by restriction

One of the early types of intensification strategy, characterized in terms of exploiting strongly determined and consistent variables in Glover [17], begins by selecting a set S as indicated for determining a penalty and incentive function, i.e. one consisting of élite solutions grouped by a clustering measure. Instead of (or in addition to) creating penalties and incentives, with the goal of incorporating attributes into the current solution that have high frequency measures over S, the method of reinforcement by restriction operates by narrowing the range of possibilities allowed for adding and dropping such attributes. For example, if $x_j = p$ has a high frequency over S for only a small number of values of p, then moves are restricted to allow x_j to take only one of these values in defining a *to-attribute.* Thus, if x_j is a 0-1 variable with a high frequency measure over S for one of its values, this value will become fixed once an admissible move exists that allows such a value assignment to be made. Other assignments may be permitted, by a variant of Aspiration by Default, if the current set of restricted alternatives is unacceptable.

Initial consideration suggests such a restriction approach offers nothing beyond the options available by penalties and incentives. However, the approach can accomplish more than this for two reasons. First, explicit restrictions can substantially accelerate the execution of choice steps by reducing the number of alternatives examined. Second, and more significantly, many problems simplify and collapse once a number of explicit restrictions are introduced, allowing structural implications to surface that permit these problems to be solved

far more readily.

Reinforcement by restriction is not limited to creating an intensification effect. Given finite time and energy to explore alternatives, imposing restrictions on some attributes allows more variations to be examined for remaining unrestricted attributes than otherwise would be possible. Thus, intensification with respect to selected elements can enhance diversification over other elements, creating a form of selective diversification. Such diversification may be contrasted with the exhaustive diversification created by the more rigid memory structures of branch and bound. In an environment where the finiteness of available search effort is dwarfed by the number of alternatives that exist to be explored exhaustively, selective diversification can make a significant contribution to effective search.

Path relinking

Path relinking is initiated by selecting two solutions x' and x'' from a collection of élite solutions produced during previous search phases. A path is then generated from x' to x'', producing a solution sequence $x' = x'(1), x'(2), \ldots, x'(r) = x''$, where $x'(i+1)$ is created from $x'(i)$ at each step by choosing a move that leaves the fewest number of moves remaining to reach x''. (A choice criterion for approximating this effect is indicated below.) Finally, once the path is completed, one or more of the solutions $x'(i)$ is selected as a solution to initiate a new search phase.

This approach provides a fundamental means for pursuing the goals of intensification and diversification when its steps are implemented to exploit strategic choice rule variations. A number of alternative moves will typically qualify to produce a next solution from $x'(i)$ by the 'fewest remaining moves' criterion, consequently allowing a variety of possible paths from x' to x''. Selecting unattractive moves relative to $c(x)$ at each step will tend to produce a final series of strongly improving moves, while selecting attractive moves will tend to produce lower quality moves at the end. (The last move, however, will be improving, or leave $c(x)$ unchanged, since x'' is a local optimum.) Thus, choosing best, worst or average moves, using an aspiration criterion to override choices in the last two cases if a sufficiently attractive solution is available, provide options that produce contrasting effects in generating the indicated sequence. (Ar-

guments exist in favour of selecting best moves at each step, and then repeating the process by interchanging x' and x''.)

The issue of an appropriate aspiration is more broadly relevant to selecting a preferred $x'(i)$ for launching a new search phase, and to terminating the sequence early. The choice of one or more solutions $x'(i)$ to launch a new search phase preferably should depend not only on $c(x'(i))$ but also on the values $c(x)$ of those solutions x that can be reached by a move from $x'(i)$. In particular, when $x'(i)$ is examined to move to $x'(i+1)$, a number of candidates for $x = x'(i+1)$ will be presented for consideration. The process additionally may be varied to allow solutions to be evaluated other than those that yield $x'(i+1)$ closer to x''.

Let $x^*(i)$ denote a neighbour of $x'(i)$ that yields a minimum $c(x)$ value during an evaluation step, excluding $x^*(i) = x'(i+1)$. (If the choice rules do not automatically eliminate the possibility $x^*(i) = x'(h)$ for $h < i$, then a simple tabu restriction can be used to do this.) Then the method selects a solution $x^*(i)$ that yields a minimum value for $c(x^*(i))$ as a new point to launch the search. If only a limited set of neighbours of $x'(i)$ are examined to identify $x^*(i)$, then a superior least cost $x'(i)$, excluding x' and x'', may be selected instead. Early termination may be elected upon encountering an $x^*(i)$ that yields $c(x^*(i)) < \text{Min}\{c(x'), c(x''), c(x'(p))\}$, where $x'(p)$ is the minimum cost $x'(h)$ for all $h \leq i$. (The procedure continues without stopping if $x'(i)$, in contrast to $x^*(i)$, yields a smaller $c(x)$ value than x' and x'', since $x'(i)$ effectively adopts the rôle of x'.)

Variation and tunnelling

A variant of the path relinking approach proposed in Glover [15] starts both endpoints x' and x'' simultaneously, producing two sequences $x' = x'(1), \ldots, x'(r)$ and $x'' = x''(1), \ldots, x''(s)$. The choices are designed to yield $x'(r) = x''(s)$, for final values of r and s. To progress toward this outcome when $x'(r) \neq x''(s)$, either $x'(r)$ is selected to create $x'(r+1)$, by the criterion of minimizing the number of moves remaining to reach $x''(s)$, or $x'(s)$ is chosen to create $x''(s+1)$, by the criterion of minimizing the number of moves remaining to reach $x'(r)$. From these options, the move is selected that produces the smallest $c(x)$ value, thus also determining which of r or s is incremented on the next step.

The path relinking approach can benefit by a tunnelling approach that allows a different neighbourhood structure to be used than in the standard search phase. In particular, it often is desirable periodically to allow moves for path relinking that would normally be excluded due to the creation of infeasibility. Such a practice is less susceptible to becoming 'lost' in an infeasible region than other ways of allowing periodic infeasibility, since feasibility evidently must be recovered by the time x'' is reached. The tunnelling effect thus created offers a chance to reach solutions that might otherwise be bypassed. In the variant that starts from both x' and x'', at least one of $x'(r)$ and $x''(s)$ may be kept feasible.

Path relinking can be organized to place greater emphasis on intensification or diversification by choosing x' and x'' to share more or fewer attributes in common. Similarly choosing x' and x'' from a clustered set of élite solutions will stimulate intensification, while choosing them from two widely separated sets will stimulate diversification.

3.3.3 Extrapolated relinking

An extension of the path relinking approach, which we call extrapolated relinking, goes beyond the path endpoint x'' (or alternatively x'), to obtain solutions that span a larger region. The ability to continue beyond this endpoint results from a method for approximating the move selection criterion specified for the standard path relinking approach, which seeks a next solution that leaves the fewest moves remaining to reach x''.

Specifically, let $A(x)$ denote the set of solution attributes in x, and let A^{drop} denote the set of solution attributes that are dropped by moves performed to reach the current solution $x'(i)$, i.e. the attributes that have served as *from-attributes* in these moves. (Some of these may have been reintroduced into $x'(i)$, but they also remain in A^{drop}.) Then we seek a move at each step to maximize the number of *to-attributes* that belong to $A(x'') - A(x'(i))$, and subject to this to minimize the number that belong to $A^{drop} - A(x'')$. Such a rule can generally be implemented very efficiently, by data structures limiting the examination of moves to those containing *to-attributes* of $A(x'') - A(x'(i))$ (or permitting these moves to be examined before others).

Once $x'(r) = x''$ is reached, the process continues by modifying the choice rule as follows. The criterion now selects a move to maximize the number of its *to-attributes* not in A^{drop} minus the number of its *to-attributes* that are in A^{drop}, and subject to this, to minimize the number of its *from-attributes* that belong to $A(x'')$. (The combination of these criteria establishes an effect analogous to that achieved by the standard algebraic formula for extending a line segment beyond an endpoint. However, the secondary minimization criterion is probably less important.) The path then stops whenever no choice remains that permits the maximization criterion to be positive.

For neighbourhoods that allow relatively unrestricted choices of moves, this approach yields an extension beyond x'' that introduces new attributes, without re-incorporating any old attributes, until no move remains that satisfies this condition. The ability to go beyond the limiting points x' and x'' creates a form of diversification not available to the path that 'lies between' these points. At the same time the exterior points are influenced by the trajectory that links x' and x''.

3.3.4 Solutions evaluated but not visited

Intensification and diversification strategies may profit by the fact that a search process generates information not only about solutions actually visited, but also about additional solutions evaluated during the examination of moves not taken. One manifestation of this is exploited by reference to the solutions $x^*(i)$ in the path relinking approach.

From a different point of view, let S^* denote a subset of solutions evaluated but not visited (for example, taken from the sequence $x(1), \ldots, x(current\_iteration)$) whose elements x yield $c(x)$ values within a chosen band of attractiveness. It is relatively easy to maintain a count such as $\#S^*(\text{to } x_j = p)$, which identifies the number of times $x_j = p$ is a *to-attribute* of a trial move leading to a solution of S^*. Such a count may be differentiated further, by stipulating that the trial move must be improving, and of high quality relative to other moves examined on the same iteration. (Differentiation of this type implicitly shrinks the composition of S^*.) Then an attribute that achieves a relatively high frequency over S^*, but that has a low residence frequency over solutions actually visited, is given an incen-

tive to be incorporated into future moves, simultaneously serving the goals of both intensification and diversification. Recency and frequency interact in this approach, by disregarding the incentive if the attribute has been selected on a recent move.

3.3.5 Interval-specific penalties and incentives

A useful adjunct to the preceding ideas extends the philosophy of Aspiration by Search Direction and Aspiration by Strong Admissibility. By these aspiration criteria, improving moves are allowed to escape a tabu classification under certain conditions, but with the result of lowering their status so that they are treated as inferior improving moves.

An extension of this preserves the improving/non-improving distinction when penalties and incentives are introduced that are not intended to be pre-emptive. For this extension, evaluations are again divided into the intervals of improving and non-improving. Penalties and incentives are then given limited scope, degrading or enhancing evaluations within a given interval, but without altering the relationship between evaluations that lie in different intervals.

Incentives granted on the basis of influence are similarly made subject to this restricted shift of evaluation. Since an influential move is not usually improving in the vicinity of a local optimum, maintaining the relationship between evaluations in different intervals implies such moves will usually be selected only when no improving moves exist, other than those classified tabu. But influential moves also have a recency-based effect. Just as executing a high influence move can cancel the tabu classification of a lower influence move over a limited span of iterations, so it should reduce or cancel the incentive to select other influential moves for a corresponding duration.

3.3.6 Candidate list procedures

Section 3.2.4 stressed the importance of procedures to isolate a candidate subset of moves from a large neighbourhood, to avoid the computational expense of evaluating moves from the entire neighbourhood. Procedures of this form have been used in optimization methods for almost as long as issues of reducing computational effort have been taken seriously (since at least the 1950s and probably much

earlier). Some of the more strategic forms of these procedures came from the field of network optimization (Glover *et al.* [18], Mulvey [19], Frendewey [20]). In such approaches, the candidate subset of moves is referenced by a list that identifies their defining elements (such as indexes of variables, nodes, or arcs), and hence these approaches have acquired the name of *candidate list strategies.*

A simple form of candidate list strategy is to construct a single element list by sampling from the neighbourhood space at random, and to repeat the process if the outcome is deemed unacceptable. This is the foundation of Monte Carlo methods, as noted earlier. Studies from network optimization, however, suggest that approaches based on more systematic designs produce superior results. Generally, these involve decomposing a neighbourhood into critical subsets, and using a rule that assures subsets not examined on one iteration become scheduled for examination on subsequent iterations. For subsets appropriately determined, best outcomes result by selecting highest quality moves from these subsets, either by explicit examination of all alternatives or by using an adaptive threshold to identify such moves (see Glover *et al.* [21]).

Another kind of candidate list strategy periodically examines larger portions of the neighbourhood, creating a master list of some number of best alternatives found. The master list is then consulted to identify moves (derived from or related to those recorded) for additional iterations until a threshold of acceptability triggers the creation of a new master list.

Candidate list strategies implicitly have a diversifying influence by causing different parts of the neighbourhood space to be examined on different iterations. This suggests there may be benefit from co-ordinating such strategies with other diversification strategies, an area that remains open for investigation.

Candidate list strategies also lend themselves very naturally to parallel processing, where forms of neighbourhood decomposition otherwise examined serially are examined in parallel. Moves can be selected by choosing the best candidate from several processes, or instead each process can execute its own preferred move, generating parallel solution trajectories that are periodically co-ordinated at a higher level. These latter approaches hold considerable promise. Some of the options are described in Glover *et al.* [5].

3.3.7 Compound neighbourhoods

Identifying an effective neighbourhood for defining moves from one solution to another can be extremely important. For example, an attempt to solve a linear programming problem by choosing moves that increment or decrement problem variables, versus choosing moves that use pivot processes or directional search, obviously can make a substantial difference to the quality of the final solution obtained. The innovations that have made linear programming a powerful optimization tool rely significantly on the discovery of effective neighbourhoods for making moves.

For combinatorial applications where the possibilities for creating neighbourhoods are largely confined to various constructive or destructive processes, or to exchanges, improvements often result by combining neighbourhoods to create moves. For example, in sequencing applications such as that illustrated in Section 3.2.1, it is generally preferable to combine neighbourhoods consisting of insert moves and swap moves, allowing both types of moves to be considered at each step. Another way of combining neighbourhoods is to generate compound moves, where a sequence of simpler moves is treated as a single more complex move.

A special type of approach for creating compound moves results from a succession of steps in which an element is assigned to a new state, with the outcome of *ejecting* some other element from its current state. The ejected element is then assigned to a new state, in turn ejecting another element, and so forth, creating a chain of such operations. For example, such a process occurs in a job sequencing problem by moving a job to a new position occupied by another job, thereby ejecting this job from its position. The second job is then moved to a new position to eject another job, and so on, finally ending by inserting the last ejected job between two other jobs. This type of approach, called an *ejection chain strategy*, has useful applications for problems of many types, particularly in connection with scheduling, routing, and partitioning (Glover [22, 23], Dorndorf and Pesch [24]). A tabu search method incorporating this approach has proved highly successful for multilevel generalized assignment problems (Laguna *et al.* [25]), suggesting the relevance of ejection chain strategies for creating compound neighbourhoods in other tabu search applications.

3.3.8 Creating new attributes—vocabulary building and concept formation

A frontier area of tabu search involves the creation of new attributes out of others. The learning approach called *target analysis*, which can implicitly combine or subdivide attributes to yield a basis for improved move evaluations has been effectively used in conjunction with tabu search in scheduling applications (see Section 3.4), and provides one of the means for generating new attributes. We focus here, however, on creating new attributes by reference to a process that may be called *vocabulary building*, related to concept formation.

Vocabulary building is based on viewing a chosen set S of solutions as a text to be analyzed, by undertaking to discover attribute combinations shared in common by various solutions x in $\mathbf{X}$. Attribute combinations that emerge as significant enough to qualify as units of vocabulary, by a process to be described below, are treated as new attributes capable of being incorporated into tabu restrictions and aspiration conditions. In addition, they can be directly assembled into larger units as a basis for constructing new solutions.

We represent collections of attributes by encoding them as assignments of values to variables, which we denote by $y_j = p$, to differentiate the vector y from the vector x which possibly may have a different dimension and encoding. Normally we suppose a y vector contains enough information to be transformed into a unique x, to which it corresponds, but this assumption can be relaxed to allow more than one x to yield the same y. (It is to be noted that a specified range of different assignments for a given attribute can be expressed as a single assignment for another, which is relevant to creating vocabulary of additional utility.)

Let $Y(S)$ denote the collection of y vectors corresponding to the chosen set S of x vectors. In addition to assignments of the form $y_j = p$ which define attributes, we allow each y_j to receive the value $y_j = *$, in order to generate subvectors that identify specific attribute combinations. In particular, an attribute combination will be implicitly determined by the non-$*$ values of y.

The approach to generate vocabulary units will be to compare vectors y' and y'' by an intersection operator, $Int(y', y'')$ to yield a vector $z = Int(y', y'')$ by the rule: $z_j = y'_j$ if $y'_j = y''_j$, and $z_j = *$ if $y'_j \neq y''_j$. By this definition we also obtain $z_j = *$ if either y'_j or

$y''_j = *$. *Int* is associative, and the intersection $Int(y : y \in \mathbf{Y})$, for an arbitrary $\mathbf{Y}$, yields a z in which $z_j = y_j$ if all y_j have the same value for $y \in \mathbf{Y}$, and $z_j = *$ otherwise.

Accompanying the intersection operator, we also define a relation of containment, by the stipulation that y'' *contains* y' if $y'_j = *$ for all j such that $y'_j \neq y''_j$. Associated with this relation, we identify the *enclosure* of y' (relative to S) to be the set $Y(S : y') = \{y \in Y(S) : y$ contains $y'\}$, and define the enclosure value of y', $enc\_value(y')$, to be the number of elements in this set, i.e. the value $\#Y(S : y')$. Finally, we refer to the number of non-$*$ components of y' as the *size* of the vector, denoted $size(y')$. (If $y \in Y(S)$, the size of y is the same as its dimension.)

Clearly the greater $size(y')$ becomes, the smaller $enc\_value(y')$ tends to become. Thus for a given size s, we seek to identify vectors y' with $size(y') \geq s$ that maximize $enc\_value(y')$, and for a given enclosure value v to identify vectors y' with $enc\_value(y') \geq v$ that maximize $size(y')$. Such vectors are included among those regarded as qualifying as vocabulary units.

Similarly we include reference to weighted enclosure values, where each $y \in Y(S)$ is weighted by a measure of attractiveness (such as the value $c(x)$ of an associated solution $x \in S$), to yield $enc\_value(y')$ as a sum of the weights over $Y(S : y')$. Particular attribute values may likewise be weighted, as by a measure of influence, to yield a weighted value for $size(y')$, equal to the sum of weights over non-$*$ components of y'.

From a broader perspective, we seek vectors as vocabulary units that give rise to aggregate units called *phrases* and *sentences* with certain properties of consistency and meaning, characterized as follows. Each y_j is allowed to receive one additional value, $y_j = blank$, which may be interpreted as an empty space free to be filled by another value (in contrast to $y_j = *$, which may be interpreted as a space occupied by two conflicting values). We begin with the collection of vectors created by the intersection operator *Int*, and replace the $*$ values with *blank* values in these vectors. We then define an extended intersection operator *E_Int*, where $z = E\_Int(y', y'')$ is given by the rules defining *Int* if y'_j and y''_j are not blank. Otherwise $z_j = y'_j$ if $y''_j = blank$, and $z_j = y''_j$ if $y'_j = blank$. *E_Int* is likewise associative. The vector $z = E\_Int(y : y \in Y)$ yields $z_j = *$ if any two $y \in Y$ have

different non-*blank* values y_j, or if some y has $y_j = *$. Otherwise z_j is the common y_j value for all y with y_j non-*blank* (where $z_j = blank$ if $y_j = blank$ for all y).

The y vectors created by *E_Int* are those we call *phrases.* A *sentence* (implicitly, a complete sentence) is a phrase that has no *blank* values. We call a phrase or sentence *grammatical* (logically consistent) if it has no $*$ values. Thus grammatical sentences are y vectors lacking both blank values and $*$ values, constructed from attribute combinations (subvectors) derived from the original elements of $Y(S)$. Finally we call a grammatical sentence y *meaningful* if it corresponds to, or maps into, a feasible solution x. (Sentences that are not grammatical do not have a form that permits them to be translated into an x vector, and hence cannot be meaningful.)

The elements of $Y(S)$ are all meaningful sentences, assuming they are obtained from feasible x vectors, and the goal is to find other meaningful sentences obtained from grammatical phrases and sentences constructed as indicated. More precisely, we are interested in generating meaningful sentences (hence feasible solutions) that are not limited to those that can be obtained from $Y(S)$, but that can also be obtained by one of the following strategies:

Sentence Construction Strategies

(S1)	Translate a grammatical phrase into a sentence by filling in the blanks (by the use of neighbourhoods that incorporate constructive moves).
(S2)	Identify some set of existing meaningful sentences (e.g. derived from current feasible x vectors not in S), and identify one or more phrases, generated by *E_Int* over S, that lie in each of these sentences. Then, by a succession of moves from neighbourhoods that preserve feasibility, transform each of these sentences into new meaningful sentences that retain as much of the identified phrases as possible.
(S3)	Identify portions of existing meaningful sentences that are contained in grammatical phrases, and transform these sentences into new meaningful sentences (using feasibility preserving neighbourhoods) by seeking to incorporate additional components of the indicated phrases.

The foregoing strategies can be implemented by incorporating the same tabu search incentive and penalty mechanisms for choosing

moves indicated in previous sections. We assume in these strategies that neighbourhood operations on x vectors are directly translated into associated changes in y vectors. In the case of (S1) there is no assurance that a meaningful sentence can be achieved unless the initial phrase itself is meaningful (i.e. is contained in at least one meaningful sentence) and the constructive process is capable of generating an appropriate completion. Also, in (S3) more than one grammatical phrase can contain a given part (subvector) of a meaningful sentence, and it may be appropriate to allow the targeted phrase to change according to possibilities consistent with available moves.

Although we have described vocabulary building processes in a somewhat general form to make their range of application visible, specific instances can profit from special algorithms for linking vocabulary units into sentences that are both meaningful and attractive, in the sense of creating good $c(x)$ values. An example of this is provided by vocabulary building approaches for the travelling salesman problem described in [23], where vocabulary units can be transformed into tours by specialized shortest path procedures. A number of combinatorial optimization problems are implicit in generating good sentences by these approaches, and the derivation of effective methods for handling these problems in various settings, as in the case of the travelling salesman problem, may provide a valuable contribution to search procedures generally.

3.3.9 Strategic oscillation

The strategic oscillation approach is closely linked to the origins of tabu search, and provides an effective interplay between intensification and diversification over the intermediate to long term. Strategic oscillation operates by moving until hitting a boundary, represented by feasibility or a stage of construction, that normally would represent a point where the method would stop. Instead of stopping, however, the neighbourhood definition is extended, or the evaluation criteria for selecting moves is modified, to permit the boundary to be crossed. The approach then proceeds for a specified depth beyond the boundary, and turns around. At this point the boundary is again approached and crossed, this time from the opposite direction, proceeding to a new turning point. The process of repeatedly approaching and crossing the boundary from different directions cre-

ates a form of oscillation that gives the method its name. Control over this oscillation is established by generating modified evaluations and rules of movement, depending on the region currently navigated and the direction of search. The possibility of retracing a prior trajectory is avoided by standard tabu mechanisms.

A simple example of this approach occurs for the multidimensional knapsack problem, where values of 0-1 variables are changed from 0 to 1 until reaching the boundary of feasibility. It then continues into the infeasible region using the same type of changes, but with a modified evaluator. After a selected number of steps, direction is reversed by changing variables from 1 to 0. Evaluation criteria to drive toward improvement (or smallest disimprovement) vary according to whether the movement is from more-to-less or less-to-more feasible (or infeasible), and are accompanied by associated restrictions on admissible changes to values of variables. An implementation of such an approach by Freville and Plateau [26, 27] has generated particularly high quality solutions for multidimensional knapsack problems.

A somewhat different type of application occurs for the problem of finding an optimal spanning tree subject to inequality constraints on subsets of weighted edges. One type of strategic oscillation approach for this problem results from a constructive process of adding edges to a growing tree until it is spanning, and then continuing to add edges to cross the boundary defined by the tree construction. A different graph structure results when the current solution no longer constitutes a tree, and hence a different neighbourhood is required, yielding modified rules for selecting moves. The rules again change in order to proceed in the opposite direction, removing edges until again recovering a tree. In such problems, the effort required by different rules may make it preferable to cross a boundary to different depths on different sides. One option is to approach and retreat from the boundary while remaining on a single side, without crossing (i.e. electing a crossing of 'zero depth'). In this example, additional types of boundaries may be considered, derived from the inequality constraints.

The use of strategic oscillation in applications that alternate constructive and destructive processes can be accompanied by exchange moves that maintain the construction at a given level. A *proximate optimality principle*, which states roughly that good constructions at

one level are likely to be close to good constructions at another, motivates a strategy of applying exchanges at different levels, on either side of a target structure such as a spanning tree, to obtain refined constructions before proceeding to adjacent levels.

Finally, we remark that the boundary incorporated in strategic oscillation need not be defined in terms of feasibility or structure, but can be defined in terms of a region where the search appears to gravitate. The oscillation then consists of compelling the search to move out of this region and allowing it to return.

3.4 Tabu Search Applications

Tabu search is still in an early stage of development, with a substantial majority of its applications occurring only since 1989. However, TS methods have enjoyed successes in a variety of problem settings, as represented by the partial list shown in the table below. Scheduling provides one of the most fruitful areas for modern heuristic techniques in general and for tabu search in particular. Although the scheduling applications presented in Table 3.1 are limited to those found in the published literature (or about to appear), there are a number of studies currently in progress that deal with scheduling models corresponding to modern manufacturing systems.

One of the early applications of TS in scheduling is due to Widmer and Hertz [28], who develop a TS method for the solution of the permutation flow shop problem. This problem consists of n multiple operation jobs arriving at time zero to be processed in the same order on m continuously available machines. The processing time of a job on a given machine is fixed (deterministic) and individual operations are not pre-emptable. The objective is to find the ordering of jobs that minimizes the *makespan*—the completion time of the last job.

Widmer and Hertz use a simple insertion heuristic based on a travelling salesman analogy to the permutation flow shop problem to generate the starting ordering of the jobs. The procedure considers neighbourhoods defined by swap moves, and at each iteration the best non-tabu move is executed evaluated relative to $c(x)$. The tabu tenure is exclusively set to a value of 7 moves and the tabu restriction is based on the paired attributes (job index, position). The termination criterion is specified as a maximum number of iterations.

Table 3.1: Some applications of tabu search

Brief Description	Reference
Scheduling	
Employee scheduling	Glover & McMillan [61]
Flow shop	Widmer & Hertz [28] Taillard [30] Reeves [31]
Job shop with tooling constraints	Widmer [32]
Convoy scheduling	Bovet *et al.* [62]
Single machine scheduling	Laguna *et al.* [8]
Just-in-time scheduling	Laguna & Gonzalez-Velarde [63]
Multiple-machine weighted flow time	Barnes & Laguna [64]
Flexible-resource job shop	Daniels & Mazzola [33]
Job shop scheduling	Dell'Amico & Trubian [6]
Single machine (target analysis)	Laguna & Glover [3]
Resource scheduling	Mooney & Rardin [36]
Deadlines and setup times	Woodruff & Spearman [37]
Transportation	
Travelling salesman	Malek *et al.* [38] Glover [22]
Vehicle routing	Gendreau *et al.* [39] Osman [42] Semet & Taillard [43]
Layout and circuit design	
Quadratic assignment	Skorin-Kapov [65] Taillard [41] Chakrapani & Skorin-Kapov [9]
Electronic circuit design	Bland & Dawson [66]
Telecommunications	
Path assignment	Oliveira & Stroud [67] Anderson *et al.* [58]
Bandwidth packing	Glover & Laguna [59]
Graphs	
Clustering	Glover *et al.* [68] Hansen *et al.* [45]
Graph colouring	Hertz & de Werra [69] Hertz *et al.* [70]
Stable sets in large graphs	Friden *et al.* [71]
Maximum clique	Gendreau *et al.* [7]
Probabilistic logic and expert systems	
Maximum satisfiability	Hansen & Jaumard [11]
Probabilistic logic	Jaumard *et al.* [44]
Probabilistic logic/expert systems	Hansen *et al.* [60]
Neural networks	
Learning in an associative memory	de Werra & Hertz [56]
Nonconvex optimization problems	Beyer & Ogier [57]
Others	
Multiconstraint 0-1 knapsack	Dammeyer & Voss [10]
Large-scale controlled rounding	Kelly *et al.* [47]
General fixed charge	Sun & McKeown [72]

Computational experiments compare this TS implementation with six previously developed heuristic methods. The study examines 50 problems with n and m ranging from values of 5 to 20 where the maximum number of TS iterations is set to $n + m$. In direct competition with the best previous heuristic developed by Nawaz *et al.* [29], the TS method returns superior solutions for 58%, and matches the best solution found for 92% of the problems.

This early TS procedure does not include many of the mechanisms described in this chapter which are now established as important components of the more effective procedures. Nevertheless, the study was important for being one of the first of its type, and for disclosing the relevance of TS for scheduling, thus motivating other research to follow in this area.

The study of Taillard [30] is noteworthy in this regard, applying tabu search to the flow shop sequencing problem. This work demonstrates that tabu search obtains solutions uniformly better than the best of the classical heuristics, while investing comparable solution time. In addition, although optimality of the solutions could not be proved, by allowing sufficient CPU time Taillard's TS method found optimal solutions for every problem for which a such solution was known. Reeves [31] further improves the computational efficiency of this method by incorporating a candidate list strategy; using this approach, TS consistently outperformed a simulated annealing heuristic on a wide variety of problem instances. Another study in this area by Widmer [32] develops a TS method for the solution of an important problem in scheduling models for flexible manufacturing—the job shop scheduling problem with tooling constraints. This implementation establishes the ability of the TS approach to be adapted to handle highly complex problems, with practical features disregarded by previous studies of related problems reported in the literature.

Daniels and Mazzola [33] present a TS method for the flexible-resource flow shop scheduling problem, which generalizes the classic flow shop scheduling problem by allowing job-operation processing times to depend on the amount of resource assigned to an operation. The objective is to determine the job sequence, resource-allocation policy, and operation start times that optimize system performance. The TS method employs a nested-search strategy based on a decomposition of the problem into these three main components (job

sequencing, resource allocation, and operation start times). The procedure was tested on over 1600 problems and is reported to be extremely effective. On 480 problem instances small enough to permit optimal solutions to be identified, the TS approach obtained optimal solutions for over 70% of the test problems, while incurring an average error of 0.3% and a maximum deviation from optimality of 2.5%. On larger problems, comparisons with other heuristic procedures showed the TS method was able to find significantly superior solutions. In addition, the authors note the nested TS approach holds considerable promise for efficient implementation in a parallel processing setting.

Dell'Amico and Trubian [6] apply tabu search to the notoriously difficult job-shop scheduling problem. They develop a *bi-directional* method to find 'good' feasible starting solutions. Their procedure alternates between assigning operations at the beginning and at the end of a partial schedule, which contrasts with previous uni-directional List Scheduler algorithms. In addition to starting from a good solution, their TS procedure assigns tabu tenures that are dependent on the search state and are selected from a given range. The range is periodically revised using uniform distributions to determine new upper and lower bounds. A simple intensification strategy is used that recovers the best solution found so far and treats it as the current solution, when a given number of iterations have been performed without improving the best solution. Computational experiments with 53 benchmark problem instances show this TS method is highly robust, in contrast to previously published local search procedures for this problem. In particular, the TS method outperforms two simulated annealing methods due to van Laarhoven *et al.* [34] and Matsuo *et al.* [35] in terms of both solution quality and speed. In addition, Dell'Amico and Trubian establish new best solutions for five out of seven open problems in the literature.

Laguna and Glover [3] develop a tailored TS method for the solution of a class of single machine scheduling problems with delay penalties and setup costs. This research discloses the usefulness of target analysis as a means of integrating effective diversification strategies within tabu search. The study also establishes the importance of accounting for regional dependencies of good decision criteria. The resulting procedure obtains solutions that are uniformly as good as, or better than, the best previously known solutions over a wide va-

riety of problem instances. For large problems (with 100 jobs) the margin of superiority of the method is more dramatic. (The previously best available heuristic for this class of problems was also a TS procedure, as empirically shown by Laguna *et al.* [8].)

Mooney and Rardin [36] develop a TS procedure for a special case of the problem of assigning tasks to a single primary resource, subject to constraints resulting from the pre-assignment of secondary or auxiliary resources. Potential applications of this problem include shift-oriented production and manpower scheduling problems and course scheduling, where classrooms may be primary and instructors and students may be secondary resources. This study includes 7 variants of a basic TS procedure. These variants combine the use of deterministic and random candidate list construction, several move selection rules, and strategic oscillation. An index is created to measure the level of diversification that each variant of the method is capable of achieving. Extensive experiments with randomly-generated and real data show that the TS variants with strategic oscillation achieve high levels of diversification (as measured by the defined index) while outperforming alternative approaches. The motivation for measuring diversification levels stems from the authors' conjecture that 'an algorithm that diversifies the search must cover the search space more or less evenly'. As a result of this study, it was found that a simple iterated descent approach (see Section 3.2.3) obtained high diversification levels but performed poorly in terms of solution quality. Therefore, relatively high diversification appears to be a necessary but not a sufficient condition for finding good solutions.

Woodruff and Spearman [37] present a highly innovative TS procedure for production scheduling, addressing a general sequencing problem that includes two classes of jobs with setup times, setup costs, holding costs and deadlines. A TS method is used with insertion moves to transform one trial solution into another. Due to the presence of deadline constraints, not every sequence is feasible. However, the search path is allowed to visit infeasible solutions by a form of strategic oscillation. A candidate list is also used as a means of reducing the computational effort involved in evaluating a given neighbourhood. Diversification is achieved by introducing a parameter d into the cost function. Low values of d result in the selection of the best available move (with reference to the objective function

value) as customarily done in a deterministic tabu search, while high values result in a randomized move selection which resembles a variant of probabilistic tabu search.

The tabu list designed for this approach is based on the concept of hashing functions. The list is composed of two entries for each visited sequence, the cost and the value of a simple hashing function (i.e. a value that represents the ordering of jobs in the sequence). Computational experiments were conducted on simulated data that captured the characteristics of the demand and production environment in a large circuit board plant. For a set of twenty test problems, the average deviation from optimality was 3% and optimal solutions were achieved in seventeen cases. The best solutions were found during searches using d values other than zero, which supports the contention that long-term memory considerations become important in complex problem settings. This study also marks the first application of TS where hashing functions are used to control the tabu structures. A more detailed study on these kinds of functions and their use within the TS framework is given in Woodruff and Zemel [12].

The first parallel implementation of tabu search to appear in the literature is due to Malek *et al.* [38]. In this implementation, each child process runs a copy of a serial TS method with different parameter settings (i.e. tabu list size and tabu restrictions). After specified intervals, the child processes are halted and the main process compares their results. The main process then selects the 'best' solution found and gives it as the initial solution to all the child processes. The 'best' solution is generally the one with the least tour cost, but an alternative solution is passed if the tour has been used before. The tabu data structures are blanked every time that the child processes are temporarily stopped. This scheme requires little overhead due to interprocessor communication, and implements an intensification phase around 'good' solutions that is not easily reproduced in a serial environment. This research shows the importance of parallel computing in solving large combinatorial optimization problems, and it also illustrates one possibility for exploiting the flexibility of tabu search in this kind of environment. Joining such an approach with the use of stronger move neighbourhoods, such as those of Gendreau *et al.* [39] or of Glover [40], may be expected to yield additional improvements.

Chakrapani and Skorin-Kapov [9] present a parallel implemen-

tation on the Connection Machine CM-2 of a TS method for the quadratic assignment problem. The implementation uses n^2 processors, where n is the size of the problem. A moving gap strategy is used to vary the tabu tenure dynamically. Additional intensification and diversification are achieved via frequency-based memory. The procedure proves to be very effective in terms of solution quality. The largest problems that can currently be solved by exact methods are of size $n = 20$. The authors' method easily matches all known optimal solutions and also matches best known solutions for additional problems of size up to $n = 80$. (These solutions were obtained by a TS procedure due to Taillard [41].) In addition the study by Chakrapani and Skorin-Kapov reports new best solutions to a set of published problems of size $n = 100$. A careful implementation on the Connection Machine, a massively parallel system, proves to be extremely suitable in this context. The increase in time per iteration appears to be a logarithmic function of n. This study also offers directions for alternative implementations that may be more efficient when solving very large quadratic assignment problems.

Vehicle routing constitutes another important area with many practical applications. Several TS variants and a hybrid simulated annealing/TS approach for the vehicle routing problem under capacity and distance constraints are presented by Osman [42]. The neighbourhoods are defined using a so-called λ-interchange. The hybrid simulated annealing approach, which uses a nonmonotonic TS strategy for adjusting temperatures, improves significantly over a standard SA. The hybrid approach produces new best solutions for 7 instances in a set of 14 previously published problems. However, this approach exhibits a large variance with regard to solution quality and computational time. The pure TS methods also find 7 new best solutions to problems in the same set, and in addition they maintain a good average solution quality without excessive computational effort. The procedures developed by Osman are easily adapted to the vehicle routing problem with different vehicle sizes.

Gendreau *et al.* [39] also develop a TS procedure for vehicle routing, using a somewhat different move neighbourhood than used in [42]. Their approach is tested against the previously reigning best solution approaches in the literature, and outperforms all of them in most problems. Interestingly, in spite of the different choice of move

neighbourhoods, their results are quite closely comparable to those of Osman [42].

Semet and Taillard [43] address a difficult version of the vehicle routing problem with many complicating side conditions, including different vehicle types and sizes, different regions, and restricted delivery windows. Their outcomes improve significantly over those previously obtained for those problems, and again demonstrate the ability of tabu search to be adapted to handle diverse real world features.

One of the first TS methods to use more than one tabu list is due to Gendreau *et al.* [7], which is designed to solve the maximum clique problem in graphs. The method uses add-delete moves to define neighbourhoods for the current solutions and a tabu list to store the indexes of the vertices most recently deleted. A second list is used to record the solutions visited during a specified number of most recent iterations. The second list is always active while the first one is only consulted when 'augmenting' moves are considered (i.e. moves that increase the size of the current clique). Storing previously visited solutions as part of the tabu structure is unusual in TS methods, but was achieved in this instance due to cleverly designed data structures to exploit the neighbourhood definition. Multiple tabu lists have now become common in many TS applications.

Jaumard *et al.* [44] investigate the problem of determining the consistency of probabilities that specify whether given collections of clauses are true, with extensions to include probability intervals, conditional probabilities, and perturbations to achieve satisfiability. By integrating a tabu search approach with an exact 0-1 nonlinear programming procedure for generating columns of a master linear program, they readily solved problems with up to 140 variables and 300 clauses, approximately tripling both the number of variables and the number of clauses that could be handled by existing alternative approaches.

This work is extended in the study of Hansen *et al.* [70] to address problems arising in expert systems, as in systems for medical diagnosis. Tabu search is again embedded in a column generation scheme to determine optimal changes to sets of rules that incorporate probabilities. The combinatorial complexity of this problem comes from the fact that the number of columns grows astronomically as a function of the number of logical sentences used to define rules. This extended

study is able to generate optimal solutions for rule systems containing up to 200 sentences, significantly advancing the size of such problems that previously could be addressed.

Dammeyer and Voss [10] studied the multiconstrained 0-1 knapsack problem using a TS method that incorporates tabu restrictions based on the logical structure of the attribute sequence generated. The method is compared against an improved version of a simulated annealing method from the literature specifically designed for these problems, using a testbed of 57 problems with known optimal solutions. The TS and SA methods take comparable time on these problems, but the TS method finds optimal solutions for nearly 50% more problems than simulated annealing (44 problems versus 31). On the remaining problems, deviations from optimality with the TS method were less than 2% in all cases, and less than 1% for all problems except one. Dammeyer and Voss also note the SA method to be very sensitive to the choice of control parameters, which greatly influences the solution quality. By contrast, they found the TS parameters to be very robust. Similar differences in outcomes are established in the study of quadratic semi-assignment problems by Domschke *et al.* [46].

When publishing tabular data, the United States Bureau of the Census must sometimes round fractional data to integer values or round integer data to multiples of a pre-specified base. Data integrity can be maintained by rounding tabular data subject to additivity constraints while minimizing the overall perturbation of the data. Kelly *et al.* [47] describe a tabu search procedure with strategic oscillation for solving this NP-hard problem. A lower bound is obtained by solving a network flow programming model and the corresponding solution is used as the starting point for the procedure. Strategic oscillation plays a major rôle in this TS implementation. The oscillation in this case is around the feasibility boundary. A penalty function is used first to lead the search from the lower bound solution towards the feasible region, by linearly incrementing the penalty for an aggregated measure of constraint violation. Once the procedure reaches feasibility for the first time, the penalty oscillates within a specified period. The theoretical lower bound value obtained by network optimization (which may not be attainable by any feasible solution) is used to gauge the quality of solutions found. Experiments with 270

simulated problems yield an average deviation from this lower bound of 1.32%. In addition, for 248 three-dimensional tables provided by the United States Bureau of the Census, the deviation from the lower bound was only 0.391%.

3.5 Connections and conclusions

Relationships between tabu search and other procedures like simulated annealing and genetic algorithms provide a basis for understanding similarities and contrasts in their philosophies, and for creating potentially useful hybrid combinations of these approaches. We offer some speculation on preferable directions in this regard, and also suggest how elements of tabu search can add a useful dimension to neural network approaches.

3.5.1 Simulated annealing

The contrasts between simulated annealing and tabu search are fairly conspicuous, though undoubtedly the most prominent is the focus on exploiting memory in tabu search that is absent from simulated annealing. The introduction of this focus entails associated differences in search mechanisms, and in the elements on which they operate.

Accompanying the differences directly attributable to the focus on memory, and also magnifying them, several additional elements are fundamental for understanding the relationship between the methods. We consider three such elements in order of increasing importance.

First, tabu search emphasizes scouting successive neighbourhoods to identify moves of high quality, as by candidate list approaches of the form described in Section 3.3. This contrasts with the simulated annealing approach of randomly sampling among these moves to apply an acceptance criterion that disregards the quality of other moves available. (Such an acceptance criterion provides the sole basis for sorting the moves selected in the SA method.) The relevance of this difference in orientation is accentuated for tabu search, since its neighbourhoods include linkages based on history, and therefore yield access to information for selecting moves that is not available in neighbourhoods of the type used in simulated annealing.

Next, tabu search evaluates the relative attractiveness of moves

not only in relation to objective function change, but also in relation to factors of influence. Both types of measure are significantly affected by the differentiation among move attributes, as embodied in tabu restrictions and aspiration criteria, and in turn by relationships manifested in recency, frequency, and sequential interdependence (hence, again, involving recourse to memory). Other aspects of the state of search also affect these measures, as reflected in the altered evaluations of strategic oscillation, which depend on the direction of the current trajectory and the region visited.

Finally TS emphasizes guiding the search by reference to multiple thresholds, reflected in the tenures for tabu-active attributes and in the conditional stipulations of aspiration criteria. This may be contrasted to the simulated annealing reliance on guiding the search by reference to the single threshold implicit in the temperature parameter. The treatment of thresholds by the two methods compounds this difference between them. Tabu search varies its threshold nonmonotonically, reflecting the conception that multidirectional parameter changes are essential to adapt to different conditions, and to provide a basis for locating alternatives that might otherwise be missed. This contrasts with the simulated annealing philosophy of adhering to a temperature parameter that only changes monotonically.

Hybrids are now emerging that are taking preliminary steps to bridge some of these differences, particularly in the realm of transcending the simulated annealing reliance on a monotonic temperature parameter. A hybrid method that allows temperature to be strategically manipulated, rather than progressively diminished, has been shown to yield improved performance over standard SA approaches, as noted in the work by Osman [42]. Another hybrid method that expands the SA basis for move evaluations has also been found to perform better than standard simulated annealing in the study by Kassou [48].

Consideration of these findings invites the question of whether removing the memory scaffolding of tabu search and retaining its other features may yield a viable method in its own right. A foundation for doing this by a 'tabu thresholding method' is described by Glover [13], and is reported in a study of graph layout and design problems by Verdejo and Cunquero [49] to perform more effectively than previously best methods for these problems.

3.5.2 Genetic algorithms

Genetic algorithms offer a somewhat different set of comparisons and contrasts with tabu search. As will be described in chapter 4, GAs are based on selecting subsets (usually pairs) of solutions from a population, called parents, and combining them to produce new solutions called children. Rules of combination to yield children are based on the genetic notion of crossover, which consists of interchanging solution values of particular variables, together with occasional operations such as random value changes. Children that pass a survivability test, probabilistically biased to favor those of superior quality, are then available to be chosen as parents of the next generation. The choice of parents to be matched in each generation is based on random or biased random sampling from the population (in some parallel versions executed over separate subpopulations whose best members are periodically exchanged or shared). Genetic terminology customarily refers to solutions as chromosomes, variables as genes, and values of variables as alleles.

By means of coding conventions, the genes of genetic algorithms may be compared to attributes in tabu search, or more precisely to attributes in the form underlying the residence measures of frequency-based memory. Introducing memory in GAs to track the history of genes and their alleles over subpopulations would provide an immediate and natural way to create a hybrid with TS.

Some important differences between genes and attributes should be noted, however. Differentiation of attributes into *from* and *to* components, each having different memory functions, do not have a counterpart in genetic algorithms. This results because GAs are organized to operate without reference to moves (although, strictly speaking, combination by crossover can be viewed as a special type of move). Another distinction derives from differences in the use of coding conventions. Although an attribute change, from a state to its complement, can be encoded in a zero-one variable, such a variable does not necessarily provide a convenient or useful representation for the transformations provided by moves. Tabu restrictions and aspiration criteria handle the binary aspects of complementarity without requiring explicit reference to a zero-one x vector or two-valued functions. Adopting a similar orientation (relative to the special class of moves embodied in crossover) might yield benefits for genetic al-

gorithms in dealing with issues of genetic representation, which currently pose difficult questions (see e.g. Liepens and Vose [50]).

A domain where a genetic interpretation of tabu search ideas seems possible concerns the use of vocabulary building approaches, as described in Section 3.3. Vocabulary units may suggestively be given the alternative name of 'genetic material'. By this means, such units may be viewed as substrings of genes, created by a process that selectively extracts them to establish a substring pool. As elements are accumulated from different sources within such a pool, and progressively re-integrated into *phrases* and *sentences* by vocabulary processes, a genetic parallel may be conceived of as incorporating substring templates to guide construction of new genes.

Perhaps the use of such evolving substring pools, as opposed to the exclusive focus on parents and children, would prove useful in genetic algorithms. But there are limiting factors, since the TS processes for creating vocabulary are based on conscious and strategic reconstruction, and hence do not much resemble genetic processes. To preserve the genetic metaphor, one may imagine relying on *intelligent enzymes*, operating as special subroutines to cut out appropriate components and then recombine them according to systematic principles. If this is not stretching analogy too far, the outcome may qualify as an interesting hybrid of the GA and TS approaches.

A contrast to be noted between genetic algorithms and tabu search arises in the treatment of context, i.e. in the consideration given to structure inherent in different problem classes. For tabu search, context is fundamental, embodied in the interplay of attribute definitions and the determination of move neighbourhoods, and in the choice of conditions to define tabu restrictions. Context is also implicit in the identification of amended evaluations created in association with longer-term memory, and in the regionally-dependent neighbourhoods and evaluations of strategic oscillation.

At the opposite end of the spectrum, GA literature characteristically stresses the freedom of its rules from the influence of context. Crossover, in particular, is a *context-neutral* operation, which assumes no reliance on conditions that solutions must obey in a particular problem setting, just as genes make no reference to the environment as they follow their encoded instructions for recombination (except, perhaps, in the case of mutation). Practical application,

however, generally renders this an inconvenient assumption, making solutions of interest difficult to find. Consequently, a good deal of effort in GA implementation is devoted to developing 'special crossover' operations that compensate for the difficulties created by context, effectively re-introducing it on a case by case basis. The related branch of evolutionary algorithms does not rely on the narrower genetic orientation, and hence does not regard the provision for context as a deviation (or extra-genetic innovation). Still, within these related families of approaches, there is no rigorous dedication to exploiting context, as manifested in problem structure, and no prescription to indicate how solutions might be combined systematically to achieve such exploitation, with the exception of special problems such as the TSP (see, for instance, the discussion of the paper by Whitley *et al.* [51] in chapter 4).

The chief method by which modern genetic algorithms and their cousins handle structure is by relegating its treatment to some other method. That is, genetic algorithms combine solutions by their parent-children processes at one level, and then a descent method takes over to operate on the resulting solutions to produce new solutions. These new solutions in turn are submitted to be recombined by the GA processes. In these versions, pioneered by Mühlenbein *et al.* [52], and also advanced by Davis [53] and Ulder *et al.* [54], genetic algorithms already take the form of hybrid methods. Hence, as will be further remarked in chapter 4, there is a natural basis for marrying GA and TS procedures in such approaches. But genetic algorithms and tabu search can also be joined in a more fundamental way.

Specifically, tabu search strategies for intensification and diversification are based on the following question: how can information be extracted from a set of good solutions to help uncover additional (and better) solutions? From one point of view, GAs provide an approach for answering this question, consisting of putting solutions together and interchanging components (in some loosely defined sense, if traditional crossover is not strictly enforced). Tabu search, by contrast, seeks an answer by using processes that specifically incorporate neighbourhood structures into their design.

Augmented by historical information, neighbourhood structures are used as a basis for applying penalties and incentives to induce attributes of good solutions to become incorporated into current solu-

tions. Consequently, although it may be meaningless to interchange or otherwise incorporate a set of attributes from one solution into another in a wholesale fashion, as attempted in recombination operations, a stepwise approach to this goal through the use of neighbourhood structures is entirely practicable. This observation, formulated from a slightly different perspective in Glover [15], provides a basis for creating structured combinations of solutions that embody desired characteristics such as feasibility. The use of these structured combinations makes it possible to integrate selected subsets of solutions in any system that satisfies three basic properties. Instead of being compelled to create new types of crossover to remove deficiencies of standard operators upon being confronted by changing contexts, this approach addresses context directly and makes it an essential part of the design for generating combinations. (A related manifestation of this theme is provided by the path relinking approach of Section 3.3.) The current trend of genetic algorithms seems to be increasingly compatible to adopting such an approach, particularly in the work of Mühlenbein [55], and this could provide a basis for a significant hybrid combination of genetic algorithm and tabu search ideas. In particular, we note that Mühlenbein has likewise indicated the relevance of incorporating TS types of memory into GAs.

3.5.3 Neural networks

Neural networks have a somewhat different set of goals from tabu search, although some overlaps exist. We indicate how tabu search can be used to extend certain neural net conceptions, yielding a hybrid that may have both hardware and software implications.

The basic transferable insight from tabu search is that memory components with dimensions such as recency and frequency can increase the efficacy of a system designed to evolve toward a desired state. We suggest there may be merit in fusing neural network memory with tabu search memory. (A rudimentary acquaintance with neural network ideas is assumed.)

Recency-based considerations can be introduced from tabu search into neural networks by a time delay feedback loop from a given neuron back to itself (or from a given synapse back to itself, by the device of interposing additional neurons). This permits firing rules and synapse weights to be changed only after a certain time threshold,

determined by the length of the feedback loop. Aspiration thresholds of the form conceived in tabu search can be embodied in inputs transmitted on a secondary level, giving the ability to override the time delay for altering firing thresholds and synaptic weights. Frequency-based effects employed in tabu search may similarly be incorporated by introducing a form of cumulative averaged feedback.

Time delay feedback mechanisms for creating recency and frequency effects can also have other functions. In a problem-solving context, for example, it may be convenient to disregard one set of options to concentrate on another, while retaining the ability to recover the suppressed options after an interval. This familiar type of human activity is not a customary part of neural network design, but can be introduced by the time dependent functions previously indicated. In addition, a threshold can be created to allow a suppressed option to 'go unnoticed' if current activity levels fall in a certain range, effectively altering the interval before the option re-emerges for consideration. Neural network designs to incorporate those features may directly make use of the TS ideas that have made these elements effective in the problem-solving domain.

Tabu search strategies that introduce longer term intensification and diversification concerns are also relevant to neural network processes. As a foundation for blending these approaches, it is useful to adopt an orientation where a collection of neurons linked by synapses with various activation weights is treated as a set of attribute variables which can be assigned alternative values. Then the condition that synapse j (from a specified origin neuron to a specified destination neuron) is assigned an activation weight in interval p can be coded by the assignment $y_j = p$, where y_j is a component of an attribute vector y, as identified in the discussion of attribute creation processes in Section 3.2.5. A similar coding identifies the condition under which a neuron fires (or does not fire) to activate its associated synapses. As a neural network process evolves, a sequence of these attribute vectors is produced over time. The association between successive vectors may be imagined to operate by reference to a neighbourhood structure implicit in the neural architecture and associated connection weights. There may also be an implicit association with some (unknown) optimization problem, or a more explicit association with a known problem and set of constraints. In the latter

case, attribute assignments (neuron firings and synapse activation) can be evaluated for efficacy by transformation into a vector x, to be checked for feasibility by $x \in \mathbf{X}$. (We maintain a distinction between y and x since there may not be a one-one association between them.)

Time records identifying the quality of outcomes produced by recent firings, and identifying the frequency with which particular attribute assignments produce the highest quality firing outcomes, yield a basis for delaying changes in certain weight assignments and for encouraging changes in others. The concept of influence, in the form introduced in tabu search, should be considered in parallel with quality of outcomes.

Attribute creation and vocabulary building strategies as discussed in Section 3.3 have a significant potential for contributing to the issue of adaptive network design. An element notably lacking in neural networks at present is a systematic means to generate *concepts*, as where a chess player evolves an ability to detect and treat a particular configuration (class of positions) as a single unit. Vocabulary building yields a direct way to generate new units from existing ones. Applied to neural networks, such a process may operate to find embedded configurations of states that correspond to good firing outcomes, and assemble them into larger units. More particularly, starting with a set of previous firing states and weightings, represented by assignments in which y ranges over a set $Y(S)$, attribute creation processes can be used to identify and integrate significant components (subvectors). Copying and segregating these components permits associated neural connections to be treated as hardwired, i.e. locked in. This corresponds to treating the unit as a single new attribute. Activating the unit (as by setting $y_j = p$ for appropriate j and p) thus automatically activates the full associated system of firings. The duplication of components of y segregated from the original structure permits the 'original components' to continue to evolve without the hardwiring limitation. This occurs in the same way that created attributes in vocabulary building processes exist side by side with separate instances of the attributes that gave rise to them.

As noted in Table 3.1 of Section 3.4, elements of tabu search have already been incorporated into neural networks in the work of de Werra and Hertz [56] and Beyer and Ogier [57]. These applications, which respectively treat visual pattern identification and nonconvex

optimization, are reported significantly to reduce training times and increase the reliability of outcomes generated. In addition, TS principles also have been integrated into a special variant of neural networks making use of constructions called ghost images in [40].

The preceding observations suggest that TS concepts and strategies offer a variety of fruitful possibilities for creating hybrid methods in combination with other approaches. Beyond this, many opportunities exist to expand the frontiers of tabu search itself. We have undertaken to point out some of the areas likely to yield particular benefits. As shown in Section 3.4, TS appears to be opening the door to new advances in many settings, encompassing production scheduling, routing, design, network planning, expert systems, and a variety of other areas. Tabu search methods present opportunities for future research both in developing new applications and in creating improved methodology. The exploration of these realms may afford a chance to make a useful impact on the solution of practical combinatorial problems.

Acknowledgement

This work was supported in part by the Joint Air Force Office of Scientific Research and Office of Naval Research Contract No. F49620-90-C-0033 at the University of Colorado.

References

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

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

[3] M.Laguna and F.Glover (1992) Integrating target analysis and tabu search for improved scheduling systems. *Expert Systems with Applications: An International Journal*, (to appear).

[4] U.Faigle and W.Kern (1992) Some convergence results for probabilistic tabu search. *ORSA J. on Computing*, 4, 32-37.

[5] F.Glover, E.Taillard and D.de Werra (1993) A user's guide to tabu search. *Annals of Ops.Res.*, **41**, (to appear).

[6] M.Dell'Amico and M.Trubian (1993) Applying tabu search to the job-shop scheduling problem. *Annals of Ops.Res.*, **41**, (to appear).

[7] M.Gendreau, L.Salvail and P.Soriano (1992) Solving the maximum clique problem using a tabu search approach. *Discrete Appl.Math.*, (to appear).

[8] M.Laguna, J.W.Barnes and F.Glover (1991) Tabu search methods for a single machine scheduling problem. *J. of Intelligent Manufacturing*, **2**, 63-74.

[9] J.Chakrapani and J.Skorin-Kapov (1993) Massively parallel tabu search for the quadratic assignment problem. *Annals of Ops.Res.*, **41**, (to appear).

[10] F.Dammeyer and S.Voss (1993) Dynamic tabu list management using the reverse elimination method. *Annals of Ops.Res.*, **41**, (to appear).

[11] P.Hansen and B.Jaumard (1990) Algorithms for the maximum satisfiability problem. *Computing*, **44**, 279-303.

[12] D.L.Woodruff and E.Zemel (1993) Hashing vectors for tabu search *Annals of Ops.Res.*, **41**, (to appear).

[13] F.Glover (1992) *Simple tabu thresholding in optimization.* Graduate School of Business and Administration, University of Colorado at Boulder.

[14] J.Ryan (1992) *Depth and width of local optima.* Department of Mathematics, University of Colorado at Denver.

[15] F.Glover (1992) Tabu search for nonlinear and parametric optimization (with links to genetic algorithms) *Discrete Appl.Math.*, (to appear).

[16] J.P.Kelly, M.Laguna and F.Glover (1992) A study of diversification strategies for the quadratic assignment problem. *Computers & Ops.Res.*, (to appear).

[17] F.Glover (1977) Heuristics for integer programming using surrogate constraints. *Dec.Sci.*, **8**, 156-166.

[18] F.Glover, D.Karney, D.Klingman and A.Napier (1974) A computational study on start procedures, basis change criteria, and solution algorithms for transportation problems. *Man.Sci.*, **20**, 793-813.

[19] J.Mulvey (1978) Pivot strategies for primal simplex network codes. *J. of the ACM*, **25**, 266-270.

[20] J.Frendewey (1983) *Candidate list strategies for GN and Simplex SON methods.* Graduate School of Business and Administration, University of Colorado at Boulder.

[21] F.Glover, R.Glover and D.Klingman (1986) The threshold assignment algorithm. *Math.Prog. Study*, **26**, 12-37.

[22] F.Glover (1992) Multilevel tabu search and embedded search neighbourhoods for the travelling salesman problem. *ORSA J. on Computing.* (to appear).

[23] F.Glover (1992) *Ejection chains, reference structures, and alternating path methods for the travelling salesman problem.* Graduate School of Business and Administration, University of Colorado at Boulder.

[24] U.Dorndorf and E.Pesch (1992) *Fast clustering algorithms.* INFORM and University of Limberg.

[25] M.Laguna, J.P.Kelly, J.L.Gonzalez-Velarde and F.Glover (1991) *Tabu search for the multilevel generalized assignment problem.* Graduate School of Business and Administration, University of Colorado at Boulder.

[26] A.Freville and G.Plateau (1986) Heuristics and reduction methods for multiple constraint 0-1 linear programming problems. *EJOR*, **24**, 206-215.

[27] A.Freville and G.Plateau (1990) Hard 0-1 multiknapsack test problems for size reduction methods. *Investigacion Operativa*, **1**, 251-270.

[28] M.Widmer and A.Hertz (1989) A new heuristic method for the flow shop sequencing problem. *EJOR*, **41**, 186-193.

[29] M.Nawaz, E.E.Emscore, Jr. and I.Ham (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA*, **11**, 91-95.

[30] E.Taillard (1990) Some efficient heuristic methods for the flow shop sequencing problem. *EJOR*, **47**, 65-74.

[31] C.R.Reeves (1993) Improving the efficiency of tabu search for machine sequencing problems. *JORS*, (to appear).

[32] M.Widmer (1991) Job shop scheduling with tooling constraints: a tabu search approach. *JORS*, **42**, 75-82.

[33] R.L.Daniels and J.B.Mazzola (1993) A tabu search heuristic for the flexible-resource flow shop scheduling problem. *Annals of Ops.Res.*, **41**, (to appear).

[34] P.J.M.van Laarhoven, E.H.L.Aarts and J.K.Lenstra (1988) *Job shop scheduling by simulated annealing.* OS-R8809, Centre for Mathematics and Computer Science, Amsterdam.

[35] H.Matsuo, C.J.Suh and R.S.Sullivan (1988) *A controlled search simulated annealing method for the general job shop scheduling problem.* Graduate School of Business, The University of Texas at Austin.

[36] E.L.Mooney and R.L.Rardin (1993) Tabu search for a class of scheduling problems. *Annals of Ops.Res.*, **41**, (to appear).

[37] D.L.Woodruff and M.L.Spearman (1992) Sequencing and batching for two classes of jobs with deadlines and setup times. *J. of the Production and Operations Management Society*, (to appear).

[38] M.Malek, M.Guruswamy, M.Pandya and H.Owens (1989) Serial and parallel simulated annealing and tabu search algorithms for the travelling salesman problem. *Annals of Ops.Res.*, **21**, 59-84.

[39] M.Gendreau, A.Hertz and G.Laporte (1992) A tabu search heuristic for the vehicle routing problem. *Man.Sci.*, (to appear).

[40] F.Glover (1992) Optimization by ghost image processes in neural networks. *Computers & Ops.Res.*, (to appear).

[41] E.Taillard (1991) Robust taboo search for the quadratic assignment problem. *Parallel Computing*, **17**, 443-455.

[42] I.H.Osman (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Ops.Res.*, **41**, (to appear).

[43] F.Semet and E.Taillard (1993) Solving real-life vehicle routing problems efficiently using taboo search. *Annals of Ops.Res.*, **41**, (to appear).

[44] B.Jaumard, P.Hansen and M.Poggi di Aragao (1991) Column generation methods for probabilistic logic. *ORSA J. on Computing*, **3**, 135-148.

[45] P.Hansen, B.Jaumard and Da Silva (1992) Average linkage divisive hierarchical clustering. *J. of Classification*, (to appear).

[46] W.Domschke, P.Frost and S.Voss (1991) Tabu search techniques for the quadratic semi-assignment problem. *In* G.Fandel, T.Gulledge and A.Jones (Eds.) *New Directions for Operations Research in Manufacturing*, 389-405. Springer.

[47] J.P.Kelly, B.L.Golden and A.A.Assad (1993) Large-scale rounding using tabu search with strategic oscillation. *Annals of Ops.Res.*, **41**, (to appear).

[48] I.Kassou (1992) *Amelioration d'ordonnancements par des methodes de voisinage.* Doctoral thesis, INSA, Rouen, France.

[49] V.V.Verdejo and R.M.Cunquero (1992) *An application of the tabu thresholding techniques: minimization of the number of arcs crossing in an acyclic digraph.* Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain.

[50] G.Liepins and M.D.Vose (1990) Representational issues in genetic optimization. *J. of Experimental and Theoretical Artificial Intelligence*, **2**, 101-115.

[51] D.Whitley, T.Starkweather and D.Shaner (1991) The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination. *In* [53], 350-372.

[52] H.Mühlenbein,M.Gorges-Schleuter and O.Krämer (1988) Evolution algorithms in combinatorial optimization. *Parallel Computing*, **7**, 65-85.

[53] L.Davis (Ed.) (1991) *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

[54] N.Ulder, E.Pesch, P.J.M.van Laarhoven, H.J.Bandelt and E.H.L.Aarts (1991) Genetic local search algorithm for the travelling salesman problem. *In* R.Maenner and H.P.Schwefel (Eds.) *Parallel Problem-solving from Nature.* Lecture Notes in Computer Science 496, Springer-Verlag, 109-116.

[55] H.Mühlenbein (1992) Parallel genetic algorithms in combinatorial optimization. *In* O.Balci (Ed.) *Computer Science and Operations Research.*, Pergamon Press.)

[56] D.de Werra and A.Hertz (1989) Tabu search techniques: a tutorial and an application to neural networks. *OR Spektrum*, **11**, 131-141.

[57] D.Beyer and R.Ogier (1991) Tabu learning: a neural network search method for solving nonconvex optimization problems. *Proceedings of the International Joint Conference on Neural Networks*, IEEE and INNS, Singapore.

[58] C.A.Anderson, K.F.Jones, M.Parker and J.Ryan (1993) Path assignment for call routing: an application of tabu search. *Annals of Ops.Res.*, **41**, (to appear).

[59] F.Glover and M.Laguna (1992) Bandwidth packing: a tabu search approach. *Man.Sci.*, (to appear).

[60] P.Hansen, B.Jaumard and M.Poggi di Aragao (1992) Mixed integer column generation algorithms and the probabilistic maximum satisfiability problem. *Proc. of the 2nd Integer Programming and Combinatorial Optimization Conference*, Carnegie Mellon.

[61] F.Glover and C.McMillan (1986) The general employee scheduling problem: an integration of management science and artificial intelligence. *Computers & Ops.Res.*, **15**, 563-593.

[62] J.Bovet, C.Constantin and D.de Werra (1992) A convoy scheduling problem. *Discrete Appl.Math.*, (to appear).

[63] M.Laguna and J.L.Gonzalez-Velarde (1991) A search heuristic for just-in-time scheduling in parallel machines. *J. of Intelligent Manufacturing*, **2**, 253-260.

[64] J.W.Barnes and M.Laguna (1992) Solving the multiple-machine weighted flow time problem using tabu search. *IIE Transactions*, (to appear).

[65] J.Skorin-Kapov (1990) Tabu search applied to the quadratic assignment problem. *ORSA J. on Computing*, **2**, 33-45.

[66] J.A.Bland and G.P.Dawson (1991) Tabu search and design optimization. *Computer-Aided Design*, **23**, 195-202.

[67] S.Oliveira and G.Stroud (1989) A parallel version of tabu search and the path assignment problem. *Heuristics for Combinatorial Optimization*, **4**, 1-24.

[68] F.Glover, C.McMillan and B.Novick (1985) Interactive decision software and computer graphics for architectural and space planning. *Annals of Ops.Res.*, **5**, 557-573.

[69] A.Hertz and D.de Werra (1987) Using tabu search techniques for graph coloring. *Computing*, **29**, 345-351.

[70] A.Hertz, B.Jaumard and M.Poggi di Aragao (1992) Topology of local optima for the k-coloring problem. *Discrete Appl.Math.*, (to appear).

[71] C.Friden, A.Hertz and D.de Werra (1989) Stabulus: a technique for finding stable sets in large graphs with tabu search. *Computing*, **42**, 35-44.

[72] M.Sun and P.G.McKeown (1993) Tabu search applied to the general fixed charge problem. *Annals of Ops.Res.*, **41**, (to appear).