ELSEVIER

European Journal of Operational Research 92 (1996) 493–511

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

General local search methods

Marc Pirlot

Faculté Polytechnique de Mons, rue de Houdain 9, B-7000 Mons, Belgium

Abstract

This paper is a tutorial introduction to three recent yet widely used general heuristics: Simulated Annealing, Tabu Search, and Genetic Algorithms. A relatively precise description and an example of application are provided for each of the methods, as well as a tentative evaluation and comparison from a pragmatic point of view.

Keywords: Combinatorial optimization; Heuristic search; Local search; Simulated Annealing; Tabu Search; Genetic Algorithms; Metaheuristics

1. Introduction

This paper is an introduction to three heuristic approaches known as Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithms (GAs). These are currently intensively used to 'solve' optimization problems and, especially, combinatorial ones. As they consist of general search principles organized in a general search strategy, they may not be described as algorithms but rather as methods or meta-algorithms. Therefore, one often calls them by the terms metaheuristics or general heuristics. The names and to some extent the principles of these heuristics are originally inspired by processes or concepts which have nothing to do with optimization.

The extraordinary success of these methods (which results in a continuous flow of literature) is due to several factors: reference to optimization mechanisms in Nature (in the case of SA and GAs), general applicability of the approach, flexibility for taking into account specific constraints in real cases, excellent tradeoff between solution quality and ease of implementation or computing time.

For each of the three approaches we provide:

- a rather precise description of their basic version;
- an example of application to a selected combinatorial optimization problem;
- a short bibliography and a historic notice; and
- a glimpse of more advanced topics such as tuning of parameters, advices for strategic options, refinements of the basic ideas, and theoretical aspects (this part can be skipped in a first reading).

Our aim is to introduce the beginner to a basic version of each method and give indications about their respective strengths and weaknesses as well as about various sophistications and hybridizations. Several good texts about each of the methods have been published and the interested reader is invited to consult the literature referred to in the bibliographic notices.

* e-mail: pirlot@mathro.fpms.ac.be

PII S0377-2217(96)00007-0

In the conclusions, some considerations on the comparison of heuristics are formulated; we try to delineate domains where each of the three approaches can be fruitfully used; we advocate that each of them implement potentially useful heuristic search ideas which can be combined without taboo but also with parsimony and regard for simplicity (at least this is the author's faith). Finally some current trends in heuristic search are outlined.

The present paper is an abridged and somewhat updated version of Pirlot (1992). The reader is referred to that paper for more detail on advanced topics and for an additional chapter on neural networks in optimization.

1.1. Local search strategies

All three general heuristics we are concerned with can be considered to some extent as local search strategies; it is quite clear for SA and TS, in a more elaborate sense for GAs. Essentially, *local search* consists in moving from a solution to another one in its neighbourhood according to some well-defined rules. For definiteness, we consider the problem of minimizing a function $F(x)$ on a finite set of points X. This can be considered a general statement of a combinatorial optimization problem. A local search strategy starts from an arbitrary solution $x_1 \in X$ and at each step n, a new solution x_{n+1} is chosen in the neighbourhood $V(x_n)$ of the current solution x_n. This presupposes the definition of a *neighbourhood* structure on X; to each $x \in X$ is associated a subset $V(x) \subseteq X$ called the neighbourhood of x. For instance, if X is a set of binary vectors and $x \in X$, a neighbourhood $V(x)$ of x can be defined as the set of all solutions $x \in X$ obtained from x by flipping a single coordinate from 0 to 1 or conversely. Conventionally, we suppose that a solution never belongs to its own neighbourhood i.e. $x \notin V(x)$, $\forall x \in X$. Alternatively, one says that the neighbours of x are the solutions obtained from x by an *elementary move*. The evolution of the current solution x_n, $n = 1, 2, \ldots$, draws a *trajectory* in the search space X. The most common criterion for selecting the next solution x_{n+1} is to pick up the best one in the neighbourhood of x_n, i.e. a solution $x_{n+1} \in V(x_n)$ with

$$F(x_{n+1}) \leqslant F(x) \quad \forall x \in V(x_n).$$

Then, x_{n+1} becomes the next current solution provided it is not worse than x_n, i.e. $F(x_{n+1}) \leqslant F(x_n)$. Otherwise, the search is stopped. This strategy is usually called a *descent* or a *steepest descent* strategy. For further comparison purposes we give a formal description below. F_n^* will denote the best value of F up to step n and x_n^* is such that $F(x_n^*) = F_n^*$.

Descent Algorithm

- Initialization: select $x_1 \in X$.
- Step $n = 1, 2, \ldots$; x_n denotes the current solution.

(a) Find the best $\bar{x}$ in the neighbourhood $V(x_n)$.

(b) If $F(\bar{x}) \leqslant F(x_n)$, then $\bar{x}$ becomes the new current solution x_{n+1} at Step $n+1$ and the best value F_n^* of F (up to Step n) as well as x_n^* are updated.

(c) Else: stop.

Note that the choice of a good neighbourhood structure is generally important for the effectiveness of the process. The main weakness of the descent algorithm is its inability to escape from *local minima*. This is symbolically illustrated in Fig. 1: all solutions in the neighbourhood $V(x_n)$ are worse than x_n although, further away, there exists a global minimum of F which cannot be reached under the descent rule. Simulated Annealing and Tabu Search are local search strategies explicitly designed for avoiding such a situation. This implies temporary deterioration of the objective function.

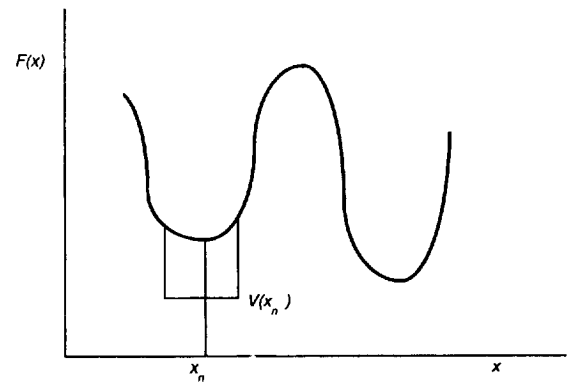

Fig. 1. Trapped in a local minimum.

2. Simulated Annealing (SA)

2.1. General presentation

In a Simulated Annealing (SA) algorithm one does not search for the best solution in the neighbourhood $V(x_n)$ of the current solution x_n; one simply draws *at random* a solution x in $V(x_n)$. If $F(x) \leqslant F(x_n)$, x becomes the next current solution. Otherwise, one of the two following alternatives is selected according to some probabilistic law; either x becomes the current solution with probability $p(n)$ or x_n remains the current solution with the complementary probability $1 - p(n)$. Typically, $p(n)$ decreases with time (n) and with the size of the deterioration of F $(= F(x) - F(x_n))$. The idea of SA originates from thermodynamics and metallurgy: when molten iron is cooled slowly enough it tends to solidify in a structure of minimal energy. This *annealing* process is mimicked by our local search strategy; at the start, almost any move (i.e. all updating of the current solution by a solution x randomly chosen in its neighbourhood) is accepted. This allows us to 'explore' the solution space. Then, gradually, the 'temperature' is decreased which means that one becomes more and more selective in accepting new solutions. By the end, only moves that improve F are accepted in practice. Schematically, SA is the following alteration of the Descent Algorithm.

Simulated Annealing (SA)

- Initialization: Select an initial solution x_1 in X; initialize the best value F^* of F and the corresponding solution x^*:

$F^* \leftarrow F(x_1)$

$x^* \leftarrow x_1$

- Step $n = 1, 2, \ldots$; x_n denotes the current solution.
  - Draw x at random in the neighbourhood $V(x_n)$ of x_n.
  - If $F(x) \leqslant F(x_n)$, then $x_{n+1} \leftarrow x$.
  - If $F(x) < F^*$, then $F^* \leftarrow F(x)$ and $x^* \leftarrow x$.
  - Else, draw a number p at random from [0, 1]. If $p \leqslant p(n)$ then $x_{n+1} \leftarrow x$.
- End: If stopping condition is fulfilled, then stop.

In order to make the algorithm operational, a few tactical decisions have to be made.

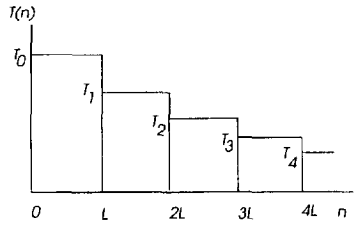

Fig. 2. The most common temperature schedule.

(a) Choice of the acceptance probability $p(n)$. On the analogy of thermodynamics, a Boltzmann-like distribution is usually chosen:

$$p(n) = \exp\left(-\frac{1}{T(n)}\Delta F_n\right),$$

where $\Delta F_n = F(x) - F(x_n)$ and $T(n)$ is the so-called 'temperature' at Step n.

(b) Choice of a 'temperature schedule' or 'cooling schedule'. Temperature $T(n)$ is a non-increasing function of time; it is designed such as to exclude almost all 'bad moves' at the end. A classical schedule is represented in Fig. 2. Starting from T_0, the temperature is maintained constant for L consecutive steps. Then, after each series of L steps, it is decreased through multiplication by a fixed factor α $(0 < \alpha < 1)$. Hence, after kL steps, the temperature sets to

$$T(kL) = T_k = \alpha^k T_0.$$

This implies the setting of three parameters, T_0, α and L, which will be respectively referred to as *initial temperature, cooling rate* and *length of plateau.*

(c) Choice of a stopping rule. Here are two variants of a stopping rule that are both natural and commonly used:

- Stop 1: If F^* was not improved by at least $\epsilon_1\%$ after K_1 consecutive series of L steps, the procedure is ended.
- Stop 2: If the number of accepted moves is less than $\epsilon_2\%$ of L for K_2 consecutive series of L steps, the procedure is ended.

Note again that the choice of a particular neighbourhood structure is a critical decision in SA as in any local search heuristic. The choice of parameter settings will be discussed in Section 3.2.

2.2. An example: Graph partitioning

We present the graph partitioning problem investigated in Johnson et al. (1989); it consists in partitioning the vertices V of the graph $G = (V, E)$ into two equal size subsets V_1 and V_2 in such a way that the number of edges with endpoints in both subsets be minimal. There are different manners of applying SA to this problem. The most natural is the following. The solution space X is the set of bipartitions (V_1, V_2) of V with $|V_1| = |V_2| = \frac{1}{2}|V|$. The objective function (to minimize) is the sum of the number of edges with both endpoints in V_1 and the number of edges with both endpoints in V_2. The neighbourhood of the current solution (V_1, V_2) is the set of all bipartitions that can be obtained by exchanging a vertex from V_1 with a vertex from V_2, i.e.

$$V_1' = V_1 \cup \{x\} \setminus \{y\},$$

$$V_2' = V_2 \cup \{y\} \setminus \{x\}$$

(for some $x \in V_2$ and $y \in V_1$). The size of the neighbourhood is $\frac{1}{4}|V|^2$. The other tactical decisions are made as explained in Section 2.1.

In an alternative implementation, instead of restricting the solution space to bipartitions in subspaces of equal cardinality (balanced bipartitions), one accepts 'illegal' solutions (V_1, V_2) with $|V_1| \neq |V_2|$. The 'solution' space is the set of *all* bipartitions and the objective function includes a term which penalizes unbalanced bipartitions:

$$F(V_1, V_2) = \text{number of edges in } V_1 + \text{number of edges in } V_2 + \gamma(|V_1| - |V_2|)^2,$$

where γ is a positive constant to be tuned. The neighbourhood of a (non-necessarily balanced) bipartition is made of all bipartitions (V_1', V_2') with

$$(V_1' = V_1 \cup \{x\} \text{ and } V_2' = V_2 \setminus \{x\})$$

or

$$(V_1' = V_1 \setminus \{y\} \text{ and } V_2' = V_2 \cup \{y\}),$$

for $x \in V_2$, $y \in V_1$. The advantage of the latter implementation which yields better results than the former, is to provide new routes for escaping from local minima and to reduce the size of the neighbourhoods.

As SA is a randomized algorithm, it is advisable to make a large number of independent runs and look for good performance on average. That is what Johnson et al. (1989) did in comparing the second implementation of SA with a local search algorithm (with the same neighbourhood) and the Kernighan–Lin specialized heuristic (see Johnson et al., 1989, and also Section 3). Roughly speaking, they conclude that SA outperforms local search; it beats Kernighan–Lin on random graphs if running time is not taken into account and works slightly better if comparable times are allocated. However, on geometric random graphs (i.e. graphs with a special kind of geometric structure, see Johnson et al., 1989), SA is outclassed by Kernighan–Lin. In all cases, the running time which is necessary for an effective annealing is usually long when compared to the time used by deterministic heuristics.

2.3. Bibliographic and historic note

The idea of applying the annealing principle to optimization problems is due to Kirkpatrick, Gelatt and Vecchi (1983) and Cerny (1985) who worked independently. They both applied SA to the traveling salesman problem. Since then, many papers have been published which either report on applications of the method in many domains or present variants and enhancements of the basic technique. A large number of theoretical papers deal with conditions under which probabilistic convergence to a global optimum can be guaranteed; some papers address the question of the speed of convergence (see, e.g. van Laarhoven and Aarts, 1987; Gidas, 1985; Mitra et al., 1986; Hajek, 1988). Extensive bibliographies can be found in Collins et al. (1988) and van Laarhoven and Aarts (1987). Several books have also been devoted to SA (see, e.g. van Laarhoven and Aarts, 1987; Azencott, 1992; Vidal, 1993; Siarry and Dreyfus, 1989). Special mention is due to a series of papers by Johnson and other authors (Johnson et al., 1989, 1991) where a very careful and methodologically sound job is done in comparing SA with other heuristics on four classical problems: graph partitioning, graph coloring and number partitioning; a third paper on the traveling salesman problem has been announced but is still awaited.

2.4. More about SA

The successes as well as the failures of SA raise important questions. In this section we touch to a selection of topics which are recurrent in the SA literature. For more detail the reader is referred to the bibliography in Section 2.3.

2.4.1. Theoretical results

A number of papers have been produced which analyse the behaviour of the annealing process as a Markov chain (either a succession of homogeneous ones or a single inhomogeneous one). Hajek (1988) gave necessary and sufficient conditions on the cooling schedule that guarantee (almost sure) convergence to an optimal solution. These conditions essentially imply that the decrease of the temperature must be logarithmic in time while in most implementations it is exponential (see the geometric decrease in Fig. 2). Using logarithmic schedules in practice results in unbearable computing times. The discrepancy between theory and practice of SA has remained an unresolved problem and one may suspect that the successes of SA have little to do with the convergence of some Markov chains (Johnson et al., 1989, report on having used a logarithmic schedule in graph partitioning without getting better results than with a geometric one).

2.4.2. Tuning the parameters

There are no generally acknowledged rules for selecting the values of the parameters even if the classical geometric cooling schedule is used. We outline some of the simplest propositions commonly found in the literature. Some experimentation (or better, systematic experimentation) around the initial choice is recommended.

Initial temperature T_0. The process should start relatively free, i.e. at sufficiently high temperature. One can, e.g. select T_0 in order to ensure a certain probability p_0 of accepting bad moves at the start; p_0 should be high, for instance 0.9, but Johnson et al. (1989) selected $p_0 = 0.4$ as the best they could find after having experimented with larger values.

Cooling factor α *and length of plateau* L. Clearly those parameters have correlated effects on the computing time. A typical value of α is 0.95. For L, one can take a multiple of the average size of a neighbourhood, in order to give a reasonable chance of trying all neighbours of the current solution when temperature is low. Johnson et al. (1989) took L equal to 16 times the size of the neighbourhood. But this is not always feasible in particular for large size problems and/or when the size of the neighbourhood is large.

Stopping criterion. Standard values (%) for ϵ_1 and ϵ_2 in the stopping criteria Stop 1 and Stop 2 (resp.) range from 1 to 5. Johnson et al. (1989) use Stop 2 with $\epsilon_2 = 2\%$ and $K_2 = 5\%$. It is advisable to control a posteriori the choice of the parameters by making sure that annealing was not stopped too early, i.e. in a phase where F_n^* was still substantially decreasing.

2.4.3. More strategic choices

Solution space, objective function and neighbourhood structure. Before tuning the parameters, some more important decisions have to be made, i.e. defining the solution space, the objective function, a neighbourhood structure. These issues are certainly crucial for a successful implementation of SA but no general recipe can be given. The best way of getting some inspiration and intuition about these strategic choices is through reading good reports on applications of SA to specific problems. Here again the papers by Johnson et al. (1989, 1991) are of high practical interest as the authors compare structurally different implementations of SA, e.g. on the graph coloring problem (approaches through penalty function, Kempe chain or fixed number of colors).

I will only add a general comment emphasizing that the three basic strategic choices are often interrelated. Both for computational and efficiency reasons it is desirable that moving through the solution space from neighbour to neighbour be as easy as possible. This is often incompatible with restricting the solution space to feasible solutions only. In graph coloring for instance, one can temporarily accept illegal colorings (i.e. in which linked vertices receive the same color).

It is very common pratice that some constraints are modelled through penalties introduced in the objective function (e.g. in Lagrangian fashion). This generally occurs in industrial applications where large numbers of specific constraints make difficult the exploration of any neighbourhood structure on the

set of feasible solutions. In most practical applications, part of the 'art' of implementing SA consists in finding a good compromise between the constraints which are modelled as soft and hard constraints, i.e. those which are modelled through penalties in the objective function and those taken into account in the definition of the solution space.

Cooling schedules. The choice of a schedule is a much discussed issue as there was a conflict, since the early days of SA, between theory (logarithmic coolings) and practice (geometric schedules). No universally valid conclusion seems to emerge from the literature. A general advice is however to cool the system slowly enough at stages where the objective function is rapidly improving. An appropriately tuned geometric schedule seems able to satisfy this requirement and yield 'good' results in a reliable manner (see, e.g. Johnson et al., 1989, pp. 882–884). However, if optimal performances are wanted, then the design of the cooling schedule can become more crucial and extremely delicate (see, e.g. van Laarhoven et al., 1992, on the jobshop scheduling problem).

Beside 'deterministic' schedules like the geometric, logarithmic or linear ones, *adaptive* schedules may also be considered. In these schedule the temperature evolution is dynamically monitored by characteristics of the trajectory in the search space. In general, this results in slowly decreasing the temperature when quick progresses are made in the objective function (see van Laarhoven and Aarts, 1987; Johnson et al., 1989, p.882).

2.4.4. Choice of the initial solution and technical improvements

Is it better to start from a randomly chosen initial solution or from a good solution, e.g. one obtained through a simple heuristic? This is another controversial question and there is again no 'yes–no' answer to it. There is an advantage in starting from a better than random solution in problems where good solutions have a special global structure that cannot be easily obtained through a small number of elementary transformations of a random solution. For instance, Johnson et al. (1989) applied their bipartitioning algorithm to graphs with a special geometric structure and found that it was beneficial to use a specially devised heuristic to build an initial solution.

Technical improvements of the SA procedure can also result in either better solutions or shorter computing times; see Johnson et al. (1989) for a few suggestions.

3. Tabu Search (TS)

Fairly general definitions of Tabu Search (TS) may be given. One can describe TS as a local search technique guided by the use of adaptive or flexible memory structures. With such a general definition, the specificity of TS is not easy to grasp. The variety of the tools and search principles introduced and described in particular by F. Glover is such that Tabu Search can be considered as the germ of a general framework for modern heuristic search. Mostly for pedagogical reasons, we restrict our presentation to the most basic features, those which are closest to the intuitive idea of 'tabu' and are mainly implemented in the so-called 'tabu lists'. In this restricted sense, TS is a method which is comparable in sophistication to SA while in the extended sense, one could even consider SA as implementing search ideas which belong to the TS arsenal. Some indications of the variations that can be built on the basic scheme are given in the concluding section of this paper. We refer to Glover (1989, 1990), Glover et al. (1993) and to Glover and Laguna (1993) for a broader presentation.

3.1. A basic short term version of Tabu Search

The adaptive memory of Tabu Search is implemented with reference to short term and long term components. Here we focus on short term aspects, though we emphasize the importance of the complementary longer term considerations in order to obtain the best computational outcomes. In the basic short term scheme, the strategy for escaping from local minima is the following one. Even if there is no better solution than the current one, x_n, in the neighbourhood $V(x_n)$, one moves to the best possible solution x in $V(x_n)$ or a sub-neighbourhood $V'(x_n) \subseteq V(x_n)$ in the case where $V(x_n)$ is too big to be explored efficiently. If the neighbourhood structure

is symmetric, i.e. if x_n belongs to the neighbourhood $V(x)$ of x whenever $x \in V(x_n)$, there is a danger of *cycling* when we explore $V(x)$ during the next step; there is indeed a chance that x_n could be the best solution in $V(x)$ in which case we would come back to x_n and from then on, oscillate between x and x_n. To avoid this situation and more general cycling situations, we could store in a list $(x_{n-1}, \ldots, x_{n-L})$ called *tabu list*, a certain number L of the last solutions encountered. If x is in the list, the move $x_n \to x$ is forbidden. This idea raises some technical problems; recording a complete description of the last visited solutions and testing for each candidate solution whether it is recorded in the list might be rather time consuming. An alternative is to record a *characteristic* or an *attribute* of the moves (it can be the transformation performed on the current solution, e.g. flipping the i-th coordinate value of a binary vector from 0 to 1).

Such a tabu list does not exactly fulfill its announced role, i.e. to prevent cycling. On the one hand it is more restrictive than necessary as it excludes many more solutions than those met in the last L iterations. On the other hand, a tabu list consisting of move characteristics can very well fail to prevent cycling even with an unbounded tabu list (fairly simple examples due to de Werra and illustrating this assertion are reported in Pirlot, 1992). This is not essential however as the main role of the tabu list is to diversify the search, i.e. to send the current solution in regions of the search space which were not previously explored and in particular to escape from local minima. An argument in favour of this thesis is that tabu search using the basic tabu list consisting of the last visited solutions (as opposed to strategically identified attributes) usually yields very poor results even if computing time is disregarded. So and despite the remarks just made, the tabu list is usually a list of one or several attributes of the recently visited solutions or (of the converse) of the most recent moves: these attributes should be chosen accurately. As mentioned above, the prohibition of solutions with a given attribute is likely to be restrictive in excluding much more solutions than the just visited ones. To correct the bad consequences of this (not all consequences are undesirable), one offers the possibility of overwriting the tabu status of a move when it leads to a good enough solution. More formally, we define an *aspiration level* that describes what is a 'good enough solution'. Two elementary examples of aspiration criteria are:

Asp 1. A solution is above the aspiration level if it is better than any solution met before (as measured by the objective function F).

Asp 2. (This criterion may be used when the tabu list records two characteristics of the moves $x \to y$, such as an attribute of the converse move $y \to x$ and the value $F(x)$.) If the attribute of the move $y' \to x'$ is in the tabu list and is associated with value F, the move is acceptable only if $F(x') < F$.

Summarizing the above, we have the following scheme for a basic short term TS algorithm.

(Short term) Tabu Search (TS)

● Initialization: Select an initial solution x_1 in X; initialize the best value F^* of F and the corresponding solution x^*:
$F^* \leftarrow F(x_1)$
$x^* \leftarrow x_1$
Tabu list TL is empty.

● Step $n = 1, 2, \ldots$; x_n denotes the current solution. $\bar{F}$ is used to store the best accessible value of F met during the exploration of the subneighbourhood $V'(x_n)$. $\bar{x}$ denotes the solution in $V'(x_n)$ for which $F(\bar{x}) = \bar{F}$. Initialize $\bar{F}$ to
$\bar{F} \leftarrow \infty$.
For all x in $V'(x_n)$,
  if $F(x) < \bar{F}$ and (if the move $(x_n \to x)$ is not tabu or if the move is tabu but passes the aspiration criterion), then
  $\bar{F} \leftarrow F(x)$ and $\bar{x} \leftarrow x$.
Let $x_{n+1} \leftarrow \bar{x}$.
If $\bar{F} < F^*$, then
  $x^* \leftarrow \bar{x}$ and $F^* \leftarrow \bar{F}$.
The appropriate characteristic of the move $(x_n \to x_{n+1})$ enters the tabu list once the first entered characteristic has been removed if the list was full.

● End: if the stopping criterion is fulfilled, then stop.

The list of tactical choices that have to be made is somewhat longer than for SA; standard choices are

not always possible or tend to be less efficient than for SA. In other words, there is more room (and need) for creativity in a TS application. The main decisions to be made are:

- the specification of a neighbourhood structure and possibly of a subneighbourhood structure (for the subneighbourhood $V'(x_n)$, a generic possibility is to pick up at random a fixed number of solutions in $V(x_n)$);
- the choice of the move attributes to be recorded in the tabu list;
- the tabu list length;
- the choice of an aspiration criterion;
- the selection of a stopping rule (the same rule as for stopping SA may be used or the total number of iterations may be fixed a priori).

3.2. Bibliographic and historic note

Glover (1986) is the initiator of Tabu Search. Similar views were developed by Hansen (1986) who formulated a steepest ascent/mildest descent heuristic principle. A comprehensive description of TS can be found in Glover et al. (1993) or in Glover and Laguna (1993) as well as extensive bibliographies. The literature about TS (mainly reports on applications) is now growing at a very fast rate. More or less refined versions of the method have been used to 'solve' a large variety of combinatorial optimization problems like scheduling, vehicle routing, quadratic assignment, clustering and grouping, electronic circuit design, graph coloring, and the traveling salesman problem. Volume 41 of the *Annals of Operations Research* (J.C. Baltzer AG, Science Publishers, Basel, Switzerland) is entirely devoted to TS. In contrast with simulated annealing very little theoretical work has been done about TS. A notable exception (Faigle and Kern, 1992) provides convergence results for a probabilistic version of TS much in the spirit of similar results for SA.

3.3. An example: Graph coloring

Because we are focusing here on the short term component of Tabu Search, we will restrict our attention to an example with a corresponding focus. An early application of TS to the coloring problem by Hertz and de Werra (1987) is appealing for its simplicity and directness. (More recent TS implementations that include additional considerations offer useful advantages and have proved more powerful, as we discuss later.) A coloring of the vertices V of a non-directed graph $G = (V, E)$ is searched for; the same color may not be assigned to two adjacent vertices and as few different colors as possible are to be used. Hertz and de Werra consider alternatively the problem of coloring the vertices with a fixed number l of colors and minimizing the number of faults, i.e. the number of adjacent vertices which are painted the same color. Hence, coloring adjacent vertices with the same color is accepted but penalized. The authors' strategy consists in finding a perfect coloring (without faults) for a large initial value of l then find successive perfect colorings for smaller and smaller l. The procedure stops when the algorithm is not able to find a coloring with the current number of allowed colors. In this implementation, a 'solution' x is any l-partition of V, $x = (V_1, \ldots, V_l)$. Clearly, in this approach, most solutions are not feasible (i.e. are not perfect colorings). This is just like in the second implementation of SA in graph partitioning. The objective function F (to minimize) is defined on all l-partitions $x = (V_1, \ldots, V_l)$ of V by

$$F(x) = \sum_{i=1}^{l} |E_i|,$$

where E_i is the set of edges whose extremities are both in V_i. The neighbourhood $V(x)$ is the set of all l-partitions which differ from $x = (V_1, \ldots, V_l)$ by the transfer of exactly one 'bad vertex' from some class V_i to some other class V_j (a 'bad vertex' in V_i is a vertex linked to another vertex of V_i). During local search, not all solutions of $V(x)$ are evaluated (there are too many of them). A sample $V'(x)$ is drawn at random from $V(x)$; the size of the subneighbourhood $V'(x)$ is a parameter of the algorithm. Only non-tabu solutions or tabu solutions that satisfy the aspiration criterion are accepted in $V'(x)$. The best solution from $V'(x)$ becomes the new current solution. The tabu list (TL) records the vertices transfered during the last k iterations together with their color before the transfer. The TL prevents a vertex transferred during the latter k iterations from turning back to its original color. The aspiration criterion is

satisfied (and the tabu status of a transfer overwritten) when a move from solution x to solution x' is such that $F(x') < F(x)$ and never in the past has a move been improving a solution of value $F(x)$ to one of value as good as $F(x')$. Note that some tricks are used to reduce computation time: the neighbourhood random generation is stopped as soon as is found a non-tabu solution which is better than the current best solution. Special star-shaped configurations are searched for and reduced, i.e. some local optimization work is done on the current solution (see Hertz and de Werra, 1987). The authors experimented on random graphs whose number of nodes ranges from 100 to 1000 and whose edge density is 0.5 (= probability of presence of each edge). They took the 'magic number 7' for the length k of their tabu list, the size of the randomly generated subneighbourhoods grows with the number of nodes of the instance at hand. The results are compared with those obtained by an implementation of SA which was run on the same instances; better results are obtained with TS and using less computing time (Johnson et al., 1991, do not confirm this statement). The same authors in the same paper also developed a so-called combined method which they use for coloring large graphs (more than 500 vertices). TS is used for finding a large independent set whose vertices are assigned the first color. These vertices are removed and a large independent set is (tabu-)searched for in the remaining part of the graph. The vertices of the second independent set receive the second color and the procedure is continued in this way until the number of not-yet-colored vertices drops below a certain threshold. Then TS is used for *coloring* the remaining vertices as explained above.

4. Genetic algorithms (GAs)

4.1. General presentation of genetic algorithms

A Genetic Algorithm (GA) may be described as a mechanism that mimics the genetic evolution of a species. The main difference with the two former approaches, SA and short term TS, is that GAs deal with *populations* of solutions rather than with single solutions. This feature is also shared with another population based approach called Scatter Search, which underlies some of the strategies of longer term TS. An obvious advantage is intrinsic parallelism but it goes beyond letting solutions evolve independently in parallel: solutions do *interact*, mix together and produce 'children' that hopefully retain the good characteristics of their parents. Population based approaches such as GAs (and Scatter Search) can be viewed as a form of local search but in a generalized sense. It is not the neighbourhood of a single solution which is explored but the *neighbourhood of a whole population*; due to interaction, this is not simply the union of the individual neighbourhoods. In GAs, the main operators used to generate and explore the neighbourhood of a population and select a new generation are *selection, crossover* and *mutation*. We describe these operators below. Note that the GA literature is rich in terms borrowed from Genetics, sometimes with a little of pedantry; we shall limit our use of the genetic jargon to a minimum. The first peculiarity of GAs is that the genetic operators do not operate directly on the solution space; solutions have to be *coded* as finite-length strings over a finite alphabet. This makes little difference with common optimization practice in some situations, like mathematical programming in 0–1 variables, as a natural encoding of a solution is a bitstring containing the values of each of the boolean variables in some predefined order. However, this is less easy in some other situations and above all, the straightforward encoding is not always the most appropriate. From now on in the framework of GAs, when we write 'solution', we mean 'encoded representation of a solution' unless otherwise stated.

A GA starts with an initial population of say N solutions and evolves it, yielding a population of the same size N at each iteration. Very roughly, the $(n+1)$st generation of solutions is obtained from the n-th generation $X^{(n)}$ through the following procedure: the best individuals (= solutions) from $X^{(n)}$ are selected, crossover operations are performed on pairs of them. This yields an offspring of solutions which will replace the bad individuals of the current population. Mutation is generally performed on a small proportion of the 'children'.

Let us go a little more into the detail. Each solution of the current population $\{x_1, \ldots, x_N\}$ is evaluated by its 'fitness' which can simply be the value of the objective function in a maximization

problem. More generally, the *fitness* of a solution is an appropriate monotone transformation of its evaluation by the objective function. Let F denote, in this section, the fitness function which we want to *maximize* over the solution space. The selection of the 'best' individuals from a given population is done according to their fitness but not in a deterministic way; solutions are drawn at random with replacement from the current population with a probability that increases with their fitness. A simple choice for such a probability is as follows: for all $i = 1, \ldots, N$, x_i is selected with probability

$$\frac{F(x_i) - F_{\min}}{\sum_{j=1}^{N}\left(F(x_j) - F_{\min}\right)},$$

where

$$F_{\min} = \min\{F(x_j),\ j = 1, \ldots, N\}.$$

Pairs of selected individuals are then submitted (with some probability χ) to the crossover operation. There are lots of possibilities for defining this operator, depending on the problem and its encoding. The commonest example called '2-point crossover' works as follows. Suppose the solutions are coded in bitstrings of length 8 and that the following pairs of individuals were selected:

0 1 0 1 1 0 0 1
1 1 0 0 0 1 1 0

Two positions, say the 2nd and the 5th, are chosen at random and the characters between those two positions are swapped, yielding two 'children' solutions. In our example, we swap the characters on the 3rd, 4th and 5th positions in both strings:

0 1 | 0 1 1 | 0 0 1
1 1 | 0 0 0 | 1 1 0

$\rightarrow$ 0 1 | 0 0 0 | 0 0 1
1 1 | 0 1 1 | 1 1 0

Each child is then submitted to mutation (with some probability μ). The simplest mutation operator consists of choosing a position at random and substituting the character in that position by another character from the alphabet. For instance, working on the 8-positions bitstring

0 1 0 1 1 0 0 1

and performing a mutation at position 6 yields

0 1 0 1 1 $\underline{1}$ 0 1

The final step in the generation of a new population is the substitution of 'bad' individuals of the current population by the (possibly mutated) children. The 'bad' individuals are selected according to their fitness in a randomized way, much as was done for selecting the good individuals; solutions are drawn at random without replacement with a probability that *decreases* with their fitness. This procedure yields the $(n+1)$st generation. The algorithm generally stops after a preassigned number of generations have been produced. As a summary, we present a schematic description of a typical GA. Note that function F (= fitness) is to be *maximized* on the space X of *coded* solutions. Many variants of this basic scheme can be found in the literature.

Genetic Algorithm (GA)

● Initialization: Select an initial population $X^{(1)} = \{x_1^{(1)}, \ldots, x_N^{(1)}\} \subseteq X$; initialize the best value F^* of F and the corresponding solution x^*:
$F^* \leftarrow \max\{F(x_i^{(1)}),\ i = 1, \ldots, N\}$
$x^* \leftarrow \arg\max\{F(x_i^{(1)}),\ i = 1, \ldots, N\}$

● Step $n = 1, 2, \ldots$; $X^{(n)}$ denotes the current population of solutions

(a) Selection of good individuals from $X^{(n)}$: Let $\{y_j,\ j = 1, \ldots, 2M\}$ be $2M$ individuals drawn with replacement from $X^{(n)}$, the probability of choosing $x_i^{(n)}$ being an increasing function of $F(x_i^{(n)})$.

(b) Crossover: For $k = 1, \ldots, M$, the crossover operator is applied to the pairs (y_{2k}, y_{2k+1}) with probability χ: this yields M pairs of children (z_{2k}, z_{2k+1}) (which are identical to their parents with probability $(1 - \chi)$).

(c) Mutation: For $j = 1, \ldots, 2M$, the mutation operator is applied to z_j with probability μ: this yields $2M$ (possibly mutated) children ω_j, $j = 1, \ldots, 2M$ (which are identical to z_j with probability $(1 - \mu)$).

(d) Substitution of the bad individuals: Draw $2M$ individuals from $X^{(n)}$ without replacement, the probability of choosing $x_i^{(n)}$ being a decreasing function of $F(x_i^{(n)})$. $X^{(n+1)}$ is obtained by substituting the $2M$ selected 'bad' individuals from $X^{(n)}$ by the children $\{z_j,\ j = 1, \ldots, 2M\}$.
For all $j = 1, \ldots, N$, if $F(x_j^{(n+1)}) > F^*$, then
$F^* \leftarrow F(x_j^{(n+1)})$
$x^* \leftarrow x_j^{(n+1)}$

● End: If $(n+1) \geqslant$ a fixed number of iterations, then stop.

Table 1
Initial population

j	Population $\{x_j^{(1)}, j=1,\ldots,4\}$	Solution	$F(x)$
1	0 1 1 0 1	13	169
2	1 1 0 0 0	24	576
3	0 1 0 0 0	8	64
4	1 0 0 1 1	19	361

4.2. A didactic example and some general remarks

We illustrate the above complicated procedure on a simple example borrowed from Goldberg (1989, p. 14 ff.). Consider the problem of maximizing $F(x) = x^2$ on the set of integers $\{0, 1, \ldots, 31\}$ and let a GA be used for this purpose.

A straightforward encoding of the solutions is by bitstrings of length 5. Let us start with the initial population of 4 solutions in Table 1: it was drawn at random by coin tossing.

In this example, the whole population is replaced by the children at each iteration (i.e. $2M = N = 4$). The good individuals selected for reproduction are shown in Table 2. Note that $x_2^{(1)}$ appears twice. A one-point crossover operator is used; all characters positioned after the selected cutpoint are swapped. The cutpoints for the two pairs of parents are shown in Table 2. The crossover probability χ is assumed to be 1 and the mutation probability is $\mu = 10^{-3}$. No mutation is simulated. The average fitness progressed from 293 in $X^{(1)}$ to 439 in $X^{(2)}$. Proceeding in this way and due to the selection mechanism designed to favour the fittest individuals it can be hoped that the final population will contain very good solutions. In the example, a very good solution (11011 = 27) is already produced after one step. It is crucial however that sufficient diversity be maintained in the population in order to permit the exploration of as many 'good regions' of the solution space as possible and not to restrict the search to the vicinity of a (local) maximum. Here again Glover's concepts of diversification and intensification which were alluded to in Section 3 are relevant. The persistence of a diversified population can be achieved by a careful tuning of the (many) parameters of the algorithm, namely population size, replacement rate (i.e. number of children substituting old solutions at each generation), crossover probability (χ) and mutation probability (μ), number of iterations.

Important structural choices have to be made as well, e.g. a good encoding for the solutions, adequate crossover and mutation operators. These are essential decisions as it is generally believed among the GAs community that the success of GAs is due to the progressive proliferation of good *schemata* in the population, i.e. specific substrings which would be associated with *properties* which *characterize* optimal or near-optimal solutions. This clearly implies that the encoding of the solutions should be in some sense *meaningful*, i.e. should implicitly give a semantic description of what is a good solution. For instance, in the above simple example, solutions with a '1' in the first position are better than any solution with a '0' in the first position. The necessity and the difficulty of selecting good encodings as well as appropriate operators is further illustrated in the next subsection. The interested reader is also referred to Falkenauer (1993) where the author proposes appropriate encoding and operators for grouping problems, including for example bin packing and clustering.

4.3. A genetic algorithm for the traveling salesman problem (TSP)

Several attempts at 'solving' the famous traveling salesman problem (TSP) have been made using different implementations of GAs. We present here the most elementary of these algorithms in order to

Table 2
First iteration

	Selected 'good' individuals $\{y_j, j=1,\ldots,2M\}$	Crossover site	Children $z_j = w_j$	Solution	$F(z_j)$
1	0 1 1 0|1	4	0 1 1 0 0	12	144
2	1 1 0 0|0	4	1 1 0 0 1	25	625
3	1 1|0 0 0	2	1 1 0 1 1	27	729
4	1 0|0 1 1	2	1 0 0 0 0	16	256

further illustrate the possible application of GAs. In the TSP, a traveller has to visit C cities exactly once and go back to his starting point. The problem (which is NP-hard) is to find a tour of the cities at lowest cost. In the simplest GA for solving the TSP, a solution, i.e. a tour, is represented by an ordered list of the cities. For instance, if there are $C = 9$ cities,

3 5 7 1 2 4 8 6 9

represents the tour that passes successively through the cities 3, 5, 7, ..., 9 and then comes back from 9 to 3. A first difficulty is to define a crossover operator since the usual 2-point crossover (see Section 4.1) is not applicable. Indeed, consider that we have to produce children from the parents

$A = 3\ 5\ 7\ 1\ 2\ 4\ 8\ 6\ 9$,

$B = 1\ 9\ 2\ 3\ 4\ 6\ 8\ 7\ 5$.

Suppose that the third and sixth positions were selected for 2-point crossover. This would yield the following 'tours' as children:

3	5	2	3	4	6	8	6	9
1	9	7	1	2	4	8	7	5

but these tours are meaningless since some cities appear twice while some other cities are not visited at all. One of the crossover operators specially designed for dealing with 'chromosomes' that represent a permutation (as is the case here) is named OX (for order crossover) and works as follows. As for 2-point crossover, two positions are selected, say the 3rd and the 6th as on the example above; both parents A and B are then prepared in order to make possible to transfer the 'genes' 3 to 6 of A in the corresponding section of B and vice versa. To prepare B for instance, holes (H) are created in the places where are located the genes that will come from A, i.e.

B : $H\ 9 | H\ 3\ H\ 6 | 8\ H\ 5$.

Then, the holes are filled in by moving non-holes that lie on their right in the chromosome. This starts from the second cut position (from the gene in the 7th position on). When the last gene is reached one goes back to the gene in the first, second, ..., position, in circular order:

B : $3\ 6 | H\ H\ H\ H | 8\ 5\ 9$.

When this process ends, there are 'holes' only in the exchange section and the genes from the exchange section of A can be imported, yielding

B' : 3 6 | 7 1 2 4 | 8 5 9.

Preparing A similarly, one gets

A' : 7 1 | 2 3 4 6 | 8 9 5.

This operator seems particularly well adapted because it introduces the least possible perturbation of the relative positions of the cities by preserving as far as possible the circular order of the cities which is what matters in this context. Other types of crossovers may also be considered for the TSP; they belong to the so-called class of 'reordering operators': see Goldberg (1989, pp. 166–179).

The problem of avoiding the creation of meaningless solutions also arises with mutation. That is why this operator is substituted by *inversion* in the TSP application. Starting from a chromosome A, two cut positions are selected at random, say before the 3rd and after the 6th sites, and the order of the cities in between is reversed, yielding

A'' : 3 5 | 4 2 1 7 | 8 6 9.

Applying these special operators in a simple-minded manner does not seem to make possible a successful treatment of problems of reasonable size. There is a need for crossover operators that do not blindly mix pairs of tours but use local optimization in the crossover operation. This is implemented in the algorithm proposed by Grefenstette et al. (1985) whose results can compete with those obtained by SA on 200-city problems. Other approaches also combining local optimization with genetic search are reported to yield near optimal results on the Padberg 532-city problem (see the references in Mathias and Whitley, 1992). This tends to demonstrate that obtaining good results with GAs on combinatorial optimization problems often requires a rather sophisticated interpretation of the basic scheme.

4.4. Bibliographic and historic note

The origins of GAs lie in the foundations of a theory of adaptive systems initiated by Holland whose 1975 book *Adaptation in Natural and Artificial Systems* is the Bible of the GA community. An introductory account of the theory as well as its main

developments and applications can be found in the excellent introduction to GAs by Goldberg (1989). Function optimization is in fact the most trivial application of the theory which ambitions to be relevant in such fields as the design of data structures, algorithms, computer operating systems, in adaptive control, etc. Since De Jong's thesis in 1975 however, a large part of the activity of the GA community has been devoted to the less exciting but more accessible task of function optimization (see Goldberg, 1989, pp. 126–127 for a list of applications of GAs to optimization problems; see also the proceedings of specialized conferences on GAs: Grefenstette, 1985, 1987; Schaffer, 1989; Belew and Booker, 1991; Schwefel, 1991; Männer and Manderick, 1992). It is interesting to note that the population based Scatter Search (SS) approach (Glover, 1977) contrasts with GAs by introducing a spatial model rather than a genetic model. In SS, reference points (parents, in GA terminology) are joined, possibly more than two at a time, by 'rounded linear combinations' instead of 'genetic crossover'. (This allows offspring to be produced beyond the region of the parents, protecting against the inbreeding phenomenon that sometimes troubles GAs.) This spatial operation produces the classic GA crossover operators and a variety of others as special instances. Additional links between different methods arise in the associated *path relinking* strategy of Tabu Search, which extends Scatter Search to combine good solutions by constructing paths in neighborhood space. These connections between SS, TS and GAs are elaborated in Glover (1994).

4.5. A few more words about GA

4.5.1. Theoretical results

GAs are not primarily designed for function optimization but as models of efficient adaptive behaviour. Hence, theoretical results are not concerned with convergence to a global optimum as for SA but with optimal or near-optimal sequences of decisions in the context of an unknown and uncertain environment. This distinction and its consequences are emphasized in a very convincing manner in De Jong (1992). The most important theoretical results about GAs are to be found in the book by Holland (1975). The author studies the proliferation of schemata (i.e. families of solutions characterized by a common substring) under simple crossover and mutation operators. He shows in particular that 'short' schemata with above average fitness proliferate at exponential rate (as long as their fitness remains above average). If we consider that the goal of heuristic search is to maximize the overall expected gain (here on the value of the objective function) and transposing results from the theory of sequential games under uncertainty, it can be argued that the GA strategy is near-optimal as it allocates exponentially growing representation to the good schemata. Such results, though conceptually appealing, are rather platonic as they do not offer any guarantee of optimality or near-optimality of the generated solutions. They only say something about the quality of the search process but perhaps stronger guarantees cannot be expected in the optimization of NP-hard problems.

4.5.2. Implementing GAs

Selection of encoding and operators. Only general comments can be made about the implementation of a GA in practice. First, it is obvious that the choice of an encoding on one hand and the design of crossover and mutation operators should be interrelated and carefully thought about. Crossover and mutation should facilitate the effective transmission of the good characters of the solutions.

Comparison with other approaches. GA practitioners have had a tendency to form a rather closed community. As a consequence there are few serious studies comparing the results of a genetic approach with other approaches on test problems. An exception is Yamada and Nakano (1992) where jobshop scheduling tests problems are tackled with success by applying a genetic search to the disjunctive graph model. Another exception is the TSP. Rather impressive results have been obtained by several authors (see, e.g. Grefenstette et al., 1985; Mühlenbein, 1991; Mathias and Whitley, 1992). This is done however by exploiting the structure of the problems and designing specific crossover operators; in one of the implementations (Mühlenbein, 1991) the genetic search is augmented by local optimization.

GAs and function optimization. The adjunction of some form of local search to a genetic approach is probably advisable in general because, as mentioned in Section 4.5.1., GAs may succeed in locating

potentially optimal regions (hopefully by detecting good schemata) but are not designed for locating precisely the optimal solution(s). This weakness can be palliated by means of local optimization or alternatively by dynamically rescaling the fitness function or using the ranking of the solutions in the current population instead of their absolute fitness. The use of local search in population approaches is a fundamental component of the original Scatter Search method, and became introduced in GAs about a decade later.

Diversity vs. convergence. A difficulty, somewhat opposite, in implementing a GA is to maintain enough diversity in the population; diversity indeed is a condition of efficiency of a crossover operator. A technique for preserving diversity consists in accepting the replacement of a solution (parent) by another (child) only when the new solution is similar to the substituted one ('niche and speciation' techniques; see Goldberg, 1989). The diversity issue is less of a complication in Scatter Search (and path relinking in TS), since the mode of combination yields an automatic mechanism to generate offspring that are not restricted to the domain of the parents.

5. Conclusion and prospects

5.1. Comparison and evaluation of heuristics

After the description of the three general heuristics presented above, the reader is naturally expecting some considerations on how they compare in applications and which to choose. It must be emphasized as a preliminary, that comparing algorithms is in general no simple matter.

Let me briefly discuss this issue. First of all, various goals may drive the design and the implementation of an algorithm. One may be wanting to solve a practical problem or trying to design the 'best' algorithm for a classical problem like the TSP. Regarding heuristics, the aim of a heuristic search may be to approximate the optimal solution and compete with exact algorithms or alternatively find a *satisfactory* solution to a practical problem. I believe that general heuristics are better suited for the former aim but can be extremely useful in the latter too. In the race for the best algorithm for a given classical problem, a current trend consists in building highly complex algorithms which incorporate knowledge and techniques from different horizons (exact optimization, heuristics, artificial intelligence, etc). Heuristics (and metaheuristics) can be used as a component of an exact algorithm, for example in producing good bounds or a good initial solution. It can be helpful too to design good implementations of general heuristics for well-solved classical problems in view of practical applications. Very often, models of real-life situations can be obtained by adding specific constraints to some classical problem; starting with a heuristic which can be evaluated against exact methods, one can take advantage of the ability of such heuristics to integrate additional constraints.

These preliminaries show that comparing general heuristics to one another and to alternative approaches essentially involves the consideration of several criteria. Among these, one should mention ease of implementation, robustness of parameter settings (w.r.t. changes in the data, instance size, etc.), flexibility for taking additional constraints into account, computational burden and solution quality. The definition of solution quality is a difficult question in itself, especially for heuristics. Beside the traditional alternative 'average versus worst case performance', there usually are no known bounds on the performance of heuristics and the evaluation relies on statistical considerations. In a worst-case analysis perspective, this raises the question of sampling the set of problem instances in a 'representative manner' or, in other terms, of how rare are the bad cases. The importance of such a question is illustrated in Johnson et al.'s (1989), study as we have already mentioned; for example, on a special class of graphs with a geometric structure, SA is outperformed by Kernighan–Lin in the graph partitioning problem. For local search heuristics, there is the additional problem of performance stability w.r.t. the initial solution choice; in randomized heuristics the choice of the random sequence is another source of variability. By the way, this shows clearly the need for a methodology in comparing algorithms; this should be based on the statistical theory of experimental design as in any other experimental science (see, e.g. Montgomery, 1976).

So, the conclusions will be in lights and shades. Let us begin with two quotations from Johnson et al.: "Annealing is a potentially valuable tool but in no

ways a panacea'' (Johnson et al., 1991, p.405). It is clear indeed that SA is not competitive in some problems (an example of which is number partitioning studied in Johnson et al., 1991). Similar statements are certainly valid both for TS and GAs. Note that a challenging problem is to understand why, or for what kind of problems, general heuristics like SA, TS, GA or others can be efficient in the search for good solutions.

The following statement from Johnson et al. (1989, p.869) is concerned with a specific difficulty in assessing *general* heuristics which are not well-specified algorithms but rather *schemes* of algorithms: ''Although experiments are capable of demonstrating that the approach performs well, it is impossible for them to prove that it performs poorly. Defenders of SA can always say that we made the wrong implementation choices''. The more complex the approach (like in TS and GAs), the more justified the statement. In particular, comparing SA, TS and GAs on a given optimization problem without specifying which implementations were used is almost meaningless. General statements should be made only when several implementations of each approach have been tested; even in that case, one should remain extremely careful. In their exemplary investigations on the applications of the SA to graph coloring, Johnson et al. (1991) compare implementations of SA using different neighbourhood structures with a classical heuristic that has been randomized (XLRF). Their conclusions indicates that the competition winner varies according to factors like instance size, edge density (for random graphs), geometry of the graph (see Johnson et al., 1991, Table VIII, p.399). Johnson et al. also experimented with the TS algorithm described in Section 3.2, but do not confirm Hertz and de Werra's conclusions: they find no *general* domination of TS over SA. A suggested explanation is that their implementation of SA is substantially faster than the SA algorithm used by Hertz and de Werra (which was originally presented in Chams et al., 1987). Note however that in most comparisons of simple TS with SA, TS has generally been found superior, mainly by being able to provide solutions of comparable quality in much shorter time.

In trying to propose some conclusions, I will be concerned only with rather straightforward (though careful) implementations of the three approaches as opposed to highly sophisticated and elaborated ones. This also means that I shall adopt the viewpoint of the practitioner who is facing a real-life problem and is trying to solve it rather than of an academic OR researcher. The latter is indeed interested in performance mainly on classical well-studied problems and simple-minded approach generally is not pertinent in this context. Note also that in recent years there has been a tendency to develop sophisticated (mainly TS) heuristics to solve practical problems. Some very nice examples can be found in Volume 41 of the *Annals of Operations Research* edited by Glover et al. (1993).

Let me first recall a common sense yet often forgotten statement. In front of a practical problem, one should inform oneself on whether good specific heuristics or exact methods do exist for the problem or for variants or subproblems. If the decision of implementing a general heuristic is made (and it can be on top of a traditional local search heuristic), I would advocate beginning with a simple one like SA or a basic short term version of TS. If encouraging results can be obtained and if better is needed, one can then turn to more sophisticated approaches, as by including longer term strategies in TS. (Experience has shown they can sometimes make dramatic differences). Indeed the time needed to implement SA or a simple TS is attractively short. For choosing further between SA and TS, I would say that it is often possible to obtain solutions of similar quality with both but TS generally runs much faster. On the contrary, even simple TS involves more tactical choices and hence needs slightly more time to be implemented and tuned. One possible option is to quickly prototype with SA and then turn to TS for more efficiency.

The quality of solution usually is better than with simple-minded descent algorithms at least for medium size, hard problems. Running times are often much longer than for descent algorithms. For large instances, they may become prohibitive and solution quality may be poor. Parameter fine tuning is not crucial in general except when trying to obtain optimal solutions (see, e.g. van Laarhoven et al., 1992). Both SA and TS implementations for a given problem can usually be adapted to take into account constraints which were not in the initial formulation of the problem.

In the case where potential benefits are expected to balance the efforts needed to sophisticating a simple implementation of TS (or SA), many possibilities are open. TS offers good opportunities for importing all kinds of heuristic search principles (including population evolution and crossover, see Moscato, 1993). As observed earlier, longer term TS already embodies population based elements in its path relinking strategies. One can also think of GAs or other heuristic approaches, or even turn back to exact methods. Incorporating knowledge about the mathematical structure of the problem is in principle advisable. A short overview of possible combinations of basic heuristic search ideas is provided in the next section.

5.2. Outlook of current trends

Beside the quality of the solutions, ease of implementation and robustness for taking additional constraints into account, a large part of the attractiveness of the so-called general heuristics is due to two of their features, the analogy with 'natural' optimization processes and the existence of theoretical convergence results. The former feature should not be taken too seriously as is usually the case with analogies, while the latter proved somewhat disappointing as discussed above.

As a consequence, nothing is taboo neither in TS nor in SA or in GAs! Many recent papers report on experimentations with variants or combinations of the 'classical' general heuristics or with new heuristic search ideas. Any sensible combination of heuristic search ideas may indeed be considered, but experimentation should be done seriously which is not always the case. It still happens that the enthusiasm raised by the intuitive appeal of a new method does not offer much resistance to pragmatic investigation.

It is likely that the field will evolve by building toolboxes of well-tried heuristic search ideas and libraries of heuristic search algorithms. In view of the almost absolute freedom of blending such tools, some people say that the design of a heuristic is an art; the blowing up of heuristic search methods will at least generate a new combinatorial optimization problem, the problem of managing the growing flow of papers submitted for publication in scientific journals.

In the final subsections we present a selection of a few trends among the most interesting in our view.

5.2.1. Sophisticated TS: A toolbox for heuristic search

In his recent work, Fred Glover gives a very extensive definition of TS which, in my opinion, is very near to a general definition of heuristic search in optimization. In the wide sense, TS can be considered a first example of a toolbox of the type alluded to above.

Following Glover, Taillard and de Werra (1993), TS is a technique based on *selected concepts of artificial intelligence* but it is 'open' in the sense that the list of concepts and search strategies usable under the TS heading is not restrictive. There is however one main component which is the implementation of *flexible memory structures*. This can be done in a variety of manners mainly by varying the neighbourhood structure and/or the objective function. We just sketch here some representative possibilities and refer to the already mentioned literature for more detail and further developments.

Tabu lists are in fact a simple way of evolving the neighboorhood structure and taking into account the history of the search. No general indications can be given on how to implement tabu lists, choosing the properties of the moves to be recorded and fixing the lists sizes. Browsing through the literature is the best way of getting inspired and, as for SA, using graphical tools and/or statistics can help a great deal to suggest fruitful choices.

As previously noted, Tabu lists are typically implementing *short term memory* effects. Longer term memory can be used for monitoring the alternation of *intensification* and *diversification* phases which are, according to Glover, a basic strategic principle of search. Intensification means concentrating the search on promising types of solutions. This can be done, e.g. by identifying characters of 'elite' solutions met in the past and defining the neighbourhood of the current solution by means of *candidate lists* of moves which implement the preference for solutions sharing some characters with elite solutions. It is clear however that identifying pertinent characters of elite solutions may not always be an easy task; alternatively properties to be emphasized in the neighbourhood definition can also be selected on the

basis of a statistical study of the search history (frequency-based memory as opposed to recency-based memory). Nevertheless, even fairly simple forms of intensification, based on identifying 'consistent' and 'strongly determined' variables, have proved highly effective (Rochat and Taillard, 1994).

Diversification is required to remain able to escape from local minima. A simple diversification device consists in modifying the objective function and penalizing solutions which are 'too close' to a set of previously visited ones. Varying the weights of such penalties as well as those introduced to model soft constraints can be used to monitor intensification and diversification ('shifting penalty approach'). A generalization of this concept is called *strategic oscillation.*

All these sophisticated search principles can better be understood through examples of applications. References to such examples are provided, e.g. in Glover et al. (1993).

5.2.2. Hybrid techniques

A number of search ideas can be imported from one type of heuristic search and implemented in another. Here is a sample of examples.

Mixing TS and SA. In an SA algorithm, random selection of a solution in the neighbourhood $V(x_n)$ may be substituted by the selection of the best solution in a randomly generated subneighbourhood $V'(x_n) \subseteq V(x_n)$. This is applied, e.g. in Tuyttens et al. (1994) where it is compared to classical SA on a grouping problem and proves definitely more efficient.

Another variant of SA can be described by means of a concept borrowed from the TS terminology. It consists in modifying the usual monotonically decreasing cooling schedule; when the search becomes inefficient, the system is heated again and the search restarted at a higher temperature; the restarting temperature is lower than the initial one and is decreased from cycle to cycle. This can be described as an application of the 'strategic oscillation' principle. For a successful implementation of this idea on the vehicle routing problem, see Osman (1993).

Conversely, the idea of a cooling schedule can be imported in TS. In many implementations of TS, a thresholding technique is used, e.g. in the aspiration criterion; moves for which the variation of the objective function does not pass a certain threshold are forbidden. The value of the threshold can be decreased according to some 'cooling schedule'. Such an idea is implemented in Tuyttens et al. (1994) where it is compared to a basic short term TS algorithm on a grouping problem. Again this yields better solutions in lesser time.

Mixing TS with GA. In an interesting paper and among other stimulating considerations about TS, Moscato (1993) proposes a parallel version of TS where a *population of solutions* is evolved in an alternation of individual and collective phases. During the collective phases, pairs of solutions are mixed by means of a crossover operator. This approach retains the most appealing idea of the GAs, i.e. interactive or co-operative parallelism. As noted, a related alternative is to incorporate Scatter Search and path relinking strategies in Tabu Search. Although proposed in TS settings some years ago, these strategies have not been widely investigated.

Integration in specific heuristics. An example of integration in a specific heuristic for graph coloring is in Hertz and de Werra (1987); it was briefly described in Section 3.3 above.

Integrating LP in general heuristics. Working with general heuristics does not preclude using more traditional techniques of OR. An example where Linear Programming is needed for computing the objective function is described in Teghem et al. (1994). The problem (grouping book covers which will be printed on the same offset plate) can be formulated as a mixed integer program; SA (or TS or GAs) may deal with the combinatorial part of the search while every assignment of covers to plates is assessed by LP. So, an LP routine is called at each step and yet computing times remain reasonable.

5.2.3. New general heuristics

Surprisingly enough, the birth rate of new general heuristic ideas is not that high. Among the recent ones let us mention for their appealing name the 'Great Deluge' and the 'Record-to-Record' algorithms (Dueck, 1993). These can be classified as rather simple thresholding techniques. Another typical example of a heuristic patterned after Nature is the so-called 'Ant algorithm' (Colorni et al., 1994) which is built on an analogy with the behavior of an insect colony.

A new approach which seems reasonably efficient has been proposed by Charon and Hudry (1993). It is called 'Noising' and relies on the following principle. Noise N is added to the objective function F for instance through random perturbations of the coefficients of F. This yields a 'noisy' objective function

$$F' = F + kN,$$

where k is a parameter. Then a descent algorithm is applied to the *noisy* objective function F'. In Variant 1, k is decreased as time elapses; Variant 2 consists of an alternation of noisy and non-noisy descents (with decreasing values of k); in Variant 3, the system periodically returns to the best solution which was met. This technique has been applied to the TSP, graph partitioning, median order and assignment problems. It is reported to be rather efficient, robust, easy to implement and to tune. Again, interconnections are possible by noting the relation to the TS approach of strategic oscillation, which is sometimes applied as a strategy to modify both objective function and constraint parameters. Similarly, periodically restarting from elite solutions is a basic TS intensification approach. Special variants of this form of intensification have recently produced unusually good outcomes in scheduling applications (Nowicki and Smutnicki, 1993, 1994).

Acknowledgments

The author wishes to warmly thank two anonymous referees for extremely constructive reports which helped preparing the revision of an initial version of the paper. In particular, I hope that their comments have led to a better balance in the presentation of the three general heuristics discussed above.

References

Azencott, R. (ed.) (1992), *Simulated Annealing. Parallelization Techniques*, Wiley, New York.

Belew, R.K., and Booker, L.B. (eds.) (1991), *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA.

Cerny, V. (1985), "A thermodynamical approach to the traveling salesman problem: An efficient simulated algorithm", *Journal of Optimization Theory and Applications* 45, 41–51.

Chams, M., Hertz, A., and de Werra, D. (1987), "Some experiments with simulated annealing for coloring graphs", *European Journal of Operational Research* 32, 260–266.

Charon, I., and Hudry, O. (1993), "La méthode du bruitage: Application au problème du voyageur de commerce", Report 93D003, TELECOM Paris.

Collins, N.E., Eglese, R.W., and Golden, B.L. (1988), "Simulated annealing: An annotated bibliography", Report No. 88-019, College of Business and Management, University of Maryland, College Park, MD.

Colorni, A., Dorigo, M., Maniezzo, V., and Trubian, M. (1994), "Ant system for Jobshop scheduling", *JORBEL. Belgian Journal of Operations Research* 34/1, 39–54.

De Jong, K.A. (1975), "An analysis of the behaviour of a class of genetic adaptive systems", doctoral Thesis, University of Michigan, Ann Arbor, MI.

De Jong, K.A. (1992), "Are genetic algorithms function optimizers?", in: R. Männer and B. Manderick, *Parallel Problem Solving from Nature, 2*, North-Holland, Amsterdam, 3–14.

Dueck, G. (1993), "New optimization heuristics. The Great Deluge algorithm and the Record-to-Record travel", *Journal of Computational Physics* 104, 86–92.

Faigle, U. and Kern, W. (1992), "Some convergence results for probabilistic Tabu Search", *ORSA Journal on Computing* 4/1, 32–37.

Falkenauer, E. (1993), "The grouping genetic algorithms: Widening the scope of the GAs", *JORBEL. Belgian Journal of Operations Research* 33, 1–2; 79–102.

Gidas, B. (1985), "Non-stationary Markov chains and convergence of the annealing algorithm", *Journal of Statistical Physics* 39, 73–131.

Glover, F. (1977), "Heuristics for integer programming using surrogate constraints", *Decision Sciences* 8/1, 156–166.

Glover, F. (1986), "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research* 5, 533–549.

Glover, F. (1989), "Tabu search – Part I, *ORSA Journal on Computing* 1, 190–206.

Glover, F. (1990), Tabu search – Part II, *ORSA Journal on Computing* 2, 4–32.

Glover, F. (1994), "Genetic Algorithms and Scatter Search: Unsuspected potentials", *Statistics and Computing* 4, 131–140.

Glover, F., and Greenberg, H.J. (1989), "New approaches for heuristic search: A bilateral linkage with artificial intelligence", *European Journal of Operational Research* 39, 119–130.

Glover, F., and Laguna, M. (1993), "Tabu Search", in: C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, Oxford, 70–150.

Glover, F., Taillard, E., and de Werra, D. (1993), "A user's guide to tabu search", *Annals of Operations Research* 41, 3–28.

Glover, F., Laguna, M., Taillard, E., and de Werra, D. (eds.) (1993), *Tabu Search*, Special Issue, *Annals of Operations Research* 41.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

Goldberg, D.E., and Lingle, R. (1985), "Alleles, loci and the

traveling salesman problem", in: J.J. Grefenstette, *Proceedings of an International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum, Hillsdale, NJ., 154–159.

Grefenstette, J.J. (ed.) (1985), *Proceedings of an International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum, Hillsdale, NJ.

Grefenstette, J.J. (ed.) (1987), *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum, Hillsdale, NJ.

Grefenstette, J.J., Gopal, R., Rosmalta, B.J., and Van Gucht, D. (1985), "Genetic algorithms for the traveling salesman problem", in: J.J. Grefenstette, *Proceedings of an International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum, Hillsdale, NJ., 160–168.

Hajek, B. (1988), "Cooling schedules for optimal annealing", *Mathematics of Operations Research* 13, 311–329.

Hansen, P. (1986), "The steepest ascent mildest descent heuristic for combinatorial programming", presented at the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

Hertz, A., and de Werra, D. (1987), "Using tabu search techniques for graph coloring", *Computing* 29, 345–351.

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1989), "Optimization by simulated annealing: An experimental evaluation – Part I (graph partitioning)", *Operations Research* 37/6, 865–892.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1991), "Optimization by simulated annealing: An experimental evaluation – Part II (graph coloring and number partitioning)", *Operations Research* 39/3, 378–406.

Kirkpatrick, S., Gelatt, Jr., C.D., and Vecchi, M.P. (1983), "Optimization by simulated annealing", *Science* 220, 671–680.

Männer, R., and Manderick, B. (1992), *Parallel Problem Solving from Nature*, 2. North-Holland, Amsterdam.

Mathias, K., and Whitley, D. (1992), "Genetic operators, the fitness landscape and the traveling salesman problem", in: R. Männer and B. Manderick, *Parallel Problem Solving from Nature*, 2. North-Holland, Amsterdam, 219–228.

Mezard, M., Parisi, G., and Virasoro, M.A. (1987), *Spin Glass Theory and Beyond*, Lecture Notes in Physics 9, World Scientific.

Montgomery, D. (1976), *Design and Analysis of Experiments*, Wiley, New York.

Moscato, P. (1993), "An introduction to population approaches for optimization and hierarchical objective function: A discussion on the role of tabu search", *Annals of Operations Research* 41, 85–122.

Mühlenbein, H. (1989), "Parallel genetic algorithms, population genetics and combinatorial optimization", in: F.D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 416–421.

Mühlenbein, H. (1991), "Parallel genetic algorithms in combinatorial optimization", to appear in: O. Balci (ed.), *Computer Science and Operations Research*, Pergamon, Oxford.

Nowicki, E., and Smutnicki, C. (1993), "A fast Taboo Search algorithm for the Job Shop problem", Report 8/93, Institute of Engineering Cybernetics, Technical University of Wroclaw; to appear in *ORSA Journal on Computing*.

Nowicki, E., and Smutnicki, C. (1994), "A fast Tabu Search algorithm for the Flow Shop problem", Institute of Engineering Cybernetics, Technical University of Wroclaw; to appear in *European Journal of Operational Research*.

Osman, I. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research* 41, 405–421.

Pirlot, M. (1992), "General local search heuristics in combinatorial optimization: A tutorial", *JORBEL. Belgian Journal of Operations Research* 32 1–2, 7–67.

Rochat, Y., and Taillard, E. (1994), "Probabilistic diversification and intensification in local search for vehicle routing", Centre de Recherche sur les Transports, University of Montreal.

Schaffer, F.D. (ed.) (1989), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA.

Schwefel, H.P. (ed.) (1991), *Parallel Problem Solving from Nature*, 1, Lecture Notes in Computer Science 496, Springer-Verlag, Berlin.

Siarry, J., and Dreyfus, G. (1989), *La Méthode du Recuit Simulé*, IDSET, Paris.

Teghem, J., Pirlot, M., and Antoniadis, C. (1994), "Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem", to appear in *Journal of Computational and Applied Mathematics*.

Tuyttens, D., Pirlot, M., Teghem, J., Trauwaert, E., and Liégeois, B. (1994), "Homogeneous grouping of nuclear fuel cans through simulated annealing and tabu search", *Annals of Operations Research* 50, 575–607.

van Laarhoven, P.J.M., and Aarts, E.H.L. (1987), *Simulated Annealing: Theory and Practice*, Kluwer Academic Publishers, Dordrecht.

van Laarhoven, P.J.M., Aarts, E.H.L., and Lenstra, J.K. (1992), "Jobshop scheduling by simulated annealing", *Operations Research* 40, 113–125.

Vidal, R.V.V. (ed.) (1992), *Applied Simulated Annealing*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin.

Yamada, T., and Nakano, R. (1992), "A genetic algorithm applicable to large-scale Jobshop problems", in: R. Männer and B. Manderick, *Parallel Problem Solving from Nature*, 2. North-Holland, Amsterdam, 281–290.