ELSEVIER

European Journal of Operational Research 135 (2001) 86–101

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

The noising methods: A generalization of some metaheuristics

Irène Charon, Olivier Hudry *

Departement d'Informatique et Réseaux, École nationale supérieure des télécommunications, 46 Rue Barrault, 75634 Paris cedex 13, France

Received 2 March 1998; accepted 6 November 2000

Abstract

In this paper, an exhaustive review of the principles and of the applications of the noising methods, recent combinatorial optimization metaheuristics, is attempted. The features and the variants of the noising methods are detailed and the tunings of their parameters when applied to different combinatorial optimization problems are summarized. The links between the noising methods and two other metaheuristics (namely, the simulated annealing method and the threshold accepting algorithm) are also studied and that the noising methods can be considered as generalizations of these metaheuristics when their components are properly chosen is shown.

Keywords: Combinatorial optimization; Metaheuristics; Local search methods; Noising methods; Simulated annealing; Threshold accepting

1. Introduction

Suppose that a company must solve daily a problem depending on data changing slightly every day, by the way of an iterative improvement method. To solve one day's problem, is it more efficient to forget what was done previously and to compute an entirely new initial solution or to restart the resolution of the problem with the solution of the day before?

Obviously, the second behaviour is faster. But, what is quite surprising is that it will lead to a better solution. It is what we observed during an experiment dealing with heuristics. We drew inspiration from this philosophy to design the *noising methods*: to compute the optimum of a combinatorial problem, instead of taking the genuine data into account directly, we consider that they are the outcomes of a series of fluctuating data converging towards the genuine ones.

The aim of this paper is to describe the possible variants of these recent combinatorial optimization methods and to review their applications and their results. These heuristics, which have been successfully applied to different combinatorial optimization problems (see Section 3), are not

* Corresponding author. Tel.: +33-1-45-81-77-77/63; fax: +33-1-45-81-31-19.

E-mail addresses: charon@infres.enst.fr (I. Charon), hudry@infres.enst.fr (O. Hudry).

0377-2217/01/$ - see front matter
PII: S0377-2217(00)00305-2

designed to solve only one special type of problems, but, as other algorithms sometimes called metaheuristics or local search methods (see for instance [1,38–40], among many references, for a general review on metaheuristics), to be applicable to various kinds of combinatorial optimization problems. Such a combinatorial optimization problem can be defined as follows: given a finite set S and a function f defined on S, find the maximum or the minimum of f over S and on element of S optimizing f. As maximizing a function f is the same as minimizing $-f$, we will assume further, without loss of generality, that we want to solve a minimization problem. In the sequel, the elements of S will be called *solutions* and f will be the *objective function*. The elements of S minimizing f over S will be called *the optimal solutions*.

In the next section, we present the principles of the noising methods. We also investigate the links between the noising methods on the one hand, and the simulated annealing method and the threshold accepting algorithm on the other hand in Section 2. Then, in Section 3, we review the applications of the noising methods to different combinatorial optimization problems and we detail some of them. Some applications were successful, some others did not give results better than those already known and reported in literature; these ones are shortly mentioned in Section 4, which contains some concluding remarks.

2. Principles of the noising methods

Many heuristics applied to combinatorial optimization problems are based on *elementary transformations*. We call *transformation* any operation which changes a solution s of S into a solution s' of S. Elementary transformations constitute a subset of the set of transformations; they usually consist in changing one feature of s without changing its global structure: for instance, if s is a binary string, an elementary transformation (e.t. in what follows) could be to change the bit located in the ith position of s into its complement, for a given i. Such an e.t. depends on one or several parameters (the parameter i for the previous example). We call *generic elementary transformation* (g.e.t. in what follows) the set of e.t.s differing one from the other just by the values taken by their parameters (the g.e.t. associated with the e.t.s of the example can be described as changing a non-specified bit into its complement). For a given g.e.t. T, the set of the solutions $t(s)$ that we can get by applying an e.t. t belonging to T (t is got by specifying the values of the parameters associated with T) to a given solution s is called the *neighbourhood* $T(s)$ of s, and the elements $t(s)$ of $T(s)$ are called the *neighbours* of s.

Thanks to such a g.e.t. T, we can design an *iterative improvement method* (also called a *descent* for a minimization problem): from the current solution s, we apply an e.t. $t \in T$ to s so that we get a neighbour $s' \in T(s)$; if we have $f(s') < f(s)$, then s' becomes the new current solution, otherwise we keep s as the current solution; then, we do it again with the current solution until it is impossible to find a neighbour of the current solution s^* which is better than $s^* : \forall s' \in T(s^*), f(s') \geqslant f(s^*)$. Thus, the solution s^* provided by a descent is a local minimum with respect to T (or similarly, with respect to the neighbourhood induced by T) and not necessarily an optimal solution.

The noising methods are also based on elementary transformations. The main difference with a descent is that, when we consider the value $f(s)$ for a given solution s, we add a perturbation that we call a *noise* to $f(s)$. This noise is randomly chosen into an interval of which the range decreases during the process. For example, if we draw the noise into the interval $[-r, +r]$ with a given probability distribution, then the noise-rate r of the noise decreases during the running of the method. Of course, it is necessary to precise the probability distribution of the noise and its extremal values (the initial – and maximum – value of r and its final – and minimum – one). It is also necessary to precise the way of decreasing the noise-rate r. Because of the random noise, it is not always sure to reach a local minimum and it can be necessary to introduce a stopping criterion, for instance by allowing a given number of transformations. Moreover, as the final solution is not necessarily the best solution computed during the method, the best solution found since the beginning is memorized and it is this best

solution which is provided at the end by the noising methods. Other variants (not exclusively linked to the noising methods) and other parameters can be useful to define the scheme of a noising method or to improve it. We detail these components below.

2.1. The generic elementary transformation

As usual for the metaheuristics, the choice of the g.e.t. (or similarly, of the neighbourhood of a solution) is essential and highly depends on the problem.

Such a g.e.t. should be able to generate the whole set S of solutions when applied enough times from any given initial solution. For example, if a solution can be described as a binary string, the g.e.t. changing a bit into its complement is relevant (this transformation is illustrated in Fig. 1(left)), but the g.e.t. changing only the bits in even positions is not. Another classic example is the g.e.t proposed by Lin [36] and often called "2-opt" for the well-known travelling salesman problem (TSP): given a Hamiltonian cycle C, remove two edges $\{x_i, x_j\}$ and $\{x_k, x_l\}$ from C and replace them by the two edges $\{x_i, x_k\}$ and $\{x_j, x_l\}$ or by $\{x_i, x_l\}$ and $\{x_k, x_j\}$ (one of these two exchanges gives another Hamiltonian cycle, the other gives a structure which is not connected). Lin's 2-opt is illustrated in Fig. 1(right). On the other hand, it should be easy and quick to evaluate the consequences involved by the transformation.

The following kinds of g.e.t.s are quite common. They are not always applicable to all combinatorial optimization problems, depending on the coding of the solutions.

- *Substitution* (or *complementation* for the solutions coded as binary strings). In the case of binary strings, the complementation transformation is illustrated in Fig. 1: a bit is changed into its complement. It can be generalized to solutions coded as strings of characters belonging to an alphabet Σ, by substituting a character of Σ to a character of the current solution, when this gives an element of S; we call *substitution* this generalization.
- *Swapping*. When the current solution is coded as a string of characters, the swapping transformation with parameters a and b consists in swapping the two characters located in positions a and b in the current solution. For instance, applying the swapping with parameters 2 and 5 to the string A **B** C D **E** F G gives A **E** C D **B** F G.
- *Shifting*. When the current solution is still coded as a string of characters, the shifting transformation with parameters a and b consists in inserting in position a the character located in position b in the current solution, and then in shifting the characters located between a (included) and b (excluded). For instance A B *C D E* **F** G becomes A B **F** *C D E* G by the shifting of parameters 3 and 6.
- *Inversion*. Once again, if the current solution is coded as a string of characters, the inversion transformation with parameters a and b consists in inverting the order of the elements located between a and b in the current solution. For example, applying an inversion with 2 and 5 as its

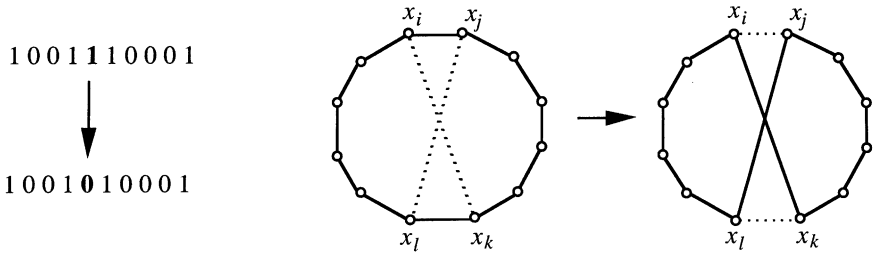

Fig. 1. Examples of elementary transformations; (left): Complementation of the fifth bit in a binary string; (right): Lin's 2-opt for the TSP applied to the edges $\{x_i, x_j\}$ and $\{x_k, x_l\}$.

parameters to A **B C D E** F G gives A **E D C B** F G. It is easy to see that Lin's 2-opt can be considered as an inversion if the Hamiltonian cycle is properly coded (as in Section 2.2).

In the following, we shall call *trial* any application of an e.t. It is the unit that we use to measure the number of iterations performed by the noising methods (or another metaheuristic) when applied to a given problem.

2.2. The examination of the neighbourhood

When the g.e.t. T (and so the neighbourhood too) is chosen, there are different ways of exploring the neighbourhood. As said above, the application of T depends on some parameters: for e.g., changing one bit in a binary string depends on one parameter: the position of the bit to be changed in the string; for the TSP, Lin's 2-opt depends on two parameters: if the current Hamiltonian cycle is $x_1\ x_2 \ldots x_n$ (where n denotes the number of vertices), the transformation specified by the two vertex-positions i and j (with $j > i+1$) consists in replacing the edges $\{x_i, x_{i+1}\}$ and $\{x_j, x_{j+1}\}$ by the edges $\{x_i, x_j\}$ and $\{x_{i+1}, x_{j+1}\}$. Then, it is possible to examine the neighbours by choosing the values of the parameters of T in different ways. Among them, we can find (see also [26] for instance):

- The *random exploration*. It consists in choosing the values of the parameters of T randomly. This strategy is applied for instance in a classic simulated annealing (see for e.g., [1,8,38–40]; this list is not exhaustive and the random exploration is widely used for simulated annealing, though it is not the only way of sampling the neighbourhood: see [40] for instance). One drawback of this strategy is that we cannot benefit by the previous trials (associated with other values of the parameters) in order to reduce the amortized complexity, thus involving generally a great amount of CPU time. Another one is that we may try a same e.t. several times (i.e., to get the same parameter-values several times) instead of exploring another part of the neighbourhood. Moreover, it is difficult to be sure to reach a local optimum. On the other hand, if the number of trials is very low, it can sample the solution-space S more efficiently than the following ways. Similarly, if the size of the neighbourhood is very large (and if we do not want to consider another g.e.t.), it can be the best way of looking for an interesting neighbour.
- The *systematic (or cyclic) exploration*. It consists in trying the different values of the parameters of T in a systematic way in order to find the first better (according to the acceptance criterion) neighbour of the current solution. For example, for the TSP if the current solution is the Hamiltonian cycle $x_1\ x_2 \ldots x_n$ (where n denotes the number of vertices), we try to apply successively a 2-opt with the parameters $(1,3)$ (since $(1,2)$ is not possible), $(1,4), \ldots,$ until $(1, n-1)$, then with $(2,4), (2,5)$, and so on until a neighbour better than the current solution is found; if no such neighbour is found before, the last trial is with the parameters $(n-2, n)$. With this kind of strategy, a neighbour is never tried twice and it is possible to be sure to have made the exploration of the whole neighbourhood. Another advantage is that, very often, the consequences of the transformation with a given set of parameters can be evaluated from the consequences of the transformation with the previous set of parameters and so we may save CPU time by reducing the amortized complexity.
- The *exhaustive exploration*. It consists in trying all the possible values of the parameters in order to keep the ones which improve the objective function the most possible. In other words, it means that we look for the best neighbour of the current solution. This strategy is applied for instance in tabu search (see [1,38–40]). The main drawback is that, when the size of the neighbourhood is quite large, it is long to explore it entirely.

Of course, it is more or less possible to mix these strategies. Hence, if the systematic exploration seems too deterministic, it is possible to start the exploration of the neighbourhood with initial parameters randomly chosen; for e.g., for the TSP, it means that instead of $(1,3)$, we can start from (i,j) with i and j randomly chosen with $j > i+1$; then $(1,3)$ will be tried after $(n-2, n)$. Similarly, we may mix the systematic exploration and the exhaustive one when the g.e.t depends on several parameters, by looking for the best values of some parameters and looking for good values for the

other parameters; for e.g., still for the TSP, we may look for the first value i for which there exists a j such that the e.t. with parameters (i,j) leads to a neighbour better than the current solution and, for this fixed value of i, we look for the best value of j. We could describe such an exploration as a "semi-exhaustive" one.

Our experimentations (see Section 3) seem to show that, for the noising methods as well as for other metaheuristics (including simulated annealing), it is more efficient or quicker to apply a systematic exploration or a kind of semi-exhaustive one.

2.3. How to add a noise

A noise is in fact a value taken by a certain random variable following a given probability distribution (e.g., a uniform law or a Gaussian one). The mean and the standard deviation of this probability distribution tend towards 0 and the standard deviation decreases during this process; in other words, the added noise progressively becomes non-existent. From a practical point of view, the noise-rate r introduced before Section 2.1 characterizes the "strength" of the noise (for a chosen probability distribution); then, the decreasing of the noise is got by decreasing r. It is possible to imagine different ways to add the noise; three of them are described below, while Section 2.4 will depict the features of the laws that have been applied in the experiments reported in this paper.

In [2–4,6,7,9–12,20–22,25,27–31,33,35,37,42], the noising methods can be considered as a succession of descents applied to the data that we perturbe before each descent. For example, suppose that the problem that we want to solve is described by a complete graph G and that $G(i,j)$ denotes the weight of the edge $\{i,j\}$. Then, one way to proceed is to replace, for any edge $\{i,j\}$, the weight $G(i,j)$ by the "noised" weight $G_{\text{noised}}(i,j) = G(i,j) + \rho(i,j)$, where $\rho(i,j)$ is a noise. Then, we apply a descent with respect to the noised data $G_{\text{noised}}(i,j)$; when a local minimum (still with respect to the noised data) is reached, we reduce the noise-rate r (in fact involving a decreasing of the standard-deviation of the noise and maybe of the absolute value of its mean), we generate new noised data $G_{\text{noised}}(i,j)$ in order to apply a new descent, and so on until the noise is low enough (it is the case for instance when the mean and the standard-deviation of the noise are equal to 0).

In [13–19,43], another noising process is designed. The genuine values of the data are always considered, but when we compute the variation $\Delta f(s,s') = f(s') - f(s)$ of the objective function f involved by the transformation of the current solution s into one of its neighbours s', instead of considering the genuine variation $\Delta f(s,s')$, we generate a noised variation $\Delta f_{\text{noised}}(s,s')$ by adding a noise to $\Delta f(s,s')$: $\Delta f_{\text{noised}}(s,s') = \Delta f(s,s') + \rho_k$, where ρ_k is a noise changing at each trial k and depending on the noise-rate. This second kind of noising generalizes the previous one, since it is always possible, for each tried e.t., to sum up the impact of the noises added to the data by a properly chosen noise added to the variation of f.

The third way is more recent: it appears in [15] (similar ideas are developed in [10]). It consists in perturbing f by forgetting a part of the data. For instance, in [18], we consider the problem of the clique partitioning of a complete weighted graph in order to minimize the sum of the weights of the edges with their two extremities in the same clique (see Section 3.1). Then, we can consider two ways to forget a part of the data. In the first one, a part of the vertices is selected randomly and the weight of an edge is taken into account in the computation of f_{noised} only if its two extremities have been selected; here, the noise-rate gives the ratio of forgotten vertices. In the second way, we do the same with the edges instead of the vertices: a part of the edges is selected and an edge is involved in the computation of f_{noised} only if it has been selected (the noise-rate still gives the ratio of forgotten data, that is, of forgotten edges). This third way of noising f can be seen as a special case of the first kind of noising: it is sufficient, for each datum (vertex or edge), to choose the added noise so that the effect of this datum is withdrawn.

There are two main differences between these three strategies. First, randomness does not occur in the same way: in the first and third possibili-

ties, after having fixed the noises affecting the genuine data, the process (a descent) can be deterministic while, in the second possibility, randomness takes place at each e.t.; in this case, the same e.t. can be rejected a first time and then accepted (or conversely), depending on the random numbers ρ_k that we draw. The second main difference is about the probability distribution. Imagine, for instance for the TSP, that the probability distribution for the random numbers $\rho(i,j)$ for the first possibility and ρ_k for the second possibility is a uniform law on the same interval $[-r,+r]$. If we apply Lin's 2-opt for the TSP, the variation of f involves four terms of the type $\rho(i,j)$ in the first kind of noising. Hence, the noise added to the variation does not follow the same probability distribution as in the second kind of noising, since the sum of four uniform random variables does not follow a uniform law. This difference between the three noising kinds is more accurate when the computation of the variation of f does not involve always the same number of terms (as it is often the case).

2.4. The probability distribution of the noise and links with some other metaheuristics

The reason for which we add a noise (to the data or to the variation) is to be able to escape a local minimum. It means that the added noise should be chosen in such a way that an e.t. yielding an increase of f (called a *bad transformation* farther) may be accepted, as in simulated annealing for instance. Usually, we add a noise chosen in an interval containing negative values as well as positive ones. The consequence is that we may accept a bad transformation, but also (contrarily to what happens with simulated annealing) we may reject an e.t. yielding a decrease of f (what we could call a *good transformation*).

In the experiments reported in Section 3, the noises are usually drawn inside an interval $[-r,+r]$, where the initial value of the noise-rate r depends on the data (for instance on the maximum weight of the graph that we deal with). Most of the time, this distribution is uniform. But, because of the phenomenon described at the end of the previous section, the noise added to the variations of f could be non-uniform and even close to a Gaussian distribution. In some experiments, another distribution has been tried, inspired by simulated annealing: the noise is given by $\rho = r \cdot \ln p$ with p uniformly drawn into]0,1[and where r is the noise-rate. We call *logarithmic* this kind of distribution.

Indeed we may consider that the second type of noising (the noise is added directly to the variations of f) is a generalization of simulated annealing if we choose the parameters properly, especially the probability distribution: in simulated annealing, the current solution s is replaced by one of its neighbours s' with a probability equal to $\min\{1, \exp(-\Delta f(s,s')/\theta)\}$, where θ is the decreasing parameter called *temperature*; then a bad transformation $(\Delta f(s,s') > 0)$ is accepted if we have: $\exp(-\Delta f(s,s')/\theta) > p$, where p is a random number uniformly drawn into]0,1[, or equivalently if the following condition is fulfilled: $\Delta f(s,s') + \theta \cdot \ln p < 0$. So, it is the same result as adding a logarithmic noise with θ as the noise-rate. This logarithmic distribution has been tried in particular in [13,14,19], as well as a uniform one.

Similarly, we may consider that the noising method is a generalization of the threshold accepting algorithm designed by Dueck and co-workers [23,24]. In this method, the neighbourhood is systematically explored and the current solution s is replaced by one of its neighbours s' if s' is better than s or if the increase (for a minimization problem) does not overpass a given threshold; this threshold depends on the iteration and decreases during the process down to 0. So, with respect to simulated annealing, the two main differences rely in the neighbourhood exploration and in the fact that the acceptance criterion is no longer the exponential and non-deterministic Metropolis criterion, but the following one:

at the kth trial, s' is accepted instead of s if

$$f(s') - f(s) < \theta_k,$$

where θ_k is the threshold of the kth trial, with $\theta_k \geqslant 0$, $\theta_k \geqslant \theta_{k+1}$ and $\theta_K = 0$ if K is the total number of trials. This criterion avoids the computation of an exponential and the call to a

random number generator, what usually saves much CPU time. One of the main difficulties of this method is the determination of the appropriate values for the thresholds θ_k, though Althöfer and Koschnik [5] have related some convergence properties of threshold accepting to those simulated annealing. It is quite easy to see that these thresholds can be seen as noises added to the variation of f (i.e., according to the second way of adding a noise described above).

One consequence of these generalizations is that they provide also results on the convergence of some noising schemes. More precisely, these convergence results (see for instance [1] for references about simulated annealing and [5] for threshold accepting algorithms) show that there exist noising schemes for which the noising methods surely converge towards an optimal solution, at least when the number of iterations is infinite. Notice also that Jacobson and Sullivan study in [32] the convergence of the first kind of noising (noises are added to the data) and give sufficient conditions for a convergence towards an optimal solution. This study is done through their approach called "generalized hill climbing algorithm", of which the aim is to provide a unifying framework to classify a large class of stochastic and deterministic algorithms, including simulated annealing, threshold accepting algorithms, genetic algorithms, tabu search and the noising methods.

To come back to the noising methods and to conclude about the probability distribution of the noise, we may of course notice that it is always possible to apply other probability distributions, for instance, Gaussian ones. Moreover, the efficiency of a distribution compared to the one of another distribution depends on the problem to solve: there is no absolutely best distribution.

2.5. Parameters of the noising methods

As for the other metaheuristics, several numerical parameters must be tuned before running a noising method. Of course, some of these parameters are linked.

- *Number of trials*. This parameter is directly linked to the CPU time that the user can (or wants to) spend to solve his/her problem. It is one stop criterion: the method is over when this number of trials is completed. Usually, the higher this number, the better the results. These trials can be gathered into clusters characterized by the fact that the parameters of the probability distribution of the noise are constant inside each cluster. It can be convenient to use the size NS of the neighbourhood induced by the adopted g.e.t. as the unit to express the total number of trials. For instance, the neighbourhood-size induced by Lin's 2-opt for the TSP is $n(n-3)/2$ if n denotes the number of vertices of the studied graph, while NS is equal to n for a binary string of n bits with the complementation as the g.e.t.
- *The noise-rate*. As stated above, the noise-rate decreases during the running of the method. The extremal values $r_{\max}$ and $r_{\min}$ of the noise-rate depend on the values of the data but it is always possible to choose $r_{\min} = 0$. If so, the last descent leads to a (at least) local minimum with respect to the adopted g.e.t. But it is often possible to save CPU time by increasing $r_{\min}$.
- *The decreasing of the noise-rate*. The decreasing of the noise-rate in the experiments reported here was usually arithmetical. In some experiments (for instance [13,14] and [19]), a geometrical decreasing was studied associated with a logarithmic noise as well as an arithmetical one associated with a uniform noise. The value of the decreasing-rate is obviously linked to the number N of trial-clusters (see above). For instance, if the noise-rate r decreases arithmetically, it decreases by $(r_{\max} - r_{\min})/N$ after each trial-cluster.

Notice that the initial solution does not seem to be an important parameter of the method. We may initialize the process with a random solution, or a solution found by another heuristic.

2.6. Automatic tuning of the parameters

As for any metaheuristic, it is not obvious to know *a priori* which parameter setting should be used. These parameters depend obviously on the problem to deal with and on the CPU time that the user wants to spend to solve his/her problem.

Anyway, an automatic tuning of the parameters of the noising methods has been designed in [19] and applied in [17–19]. The aim is to get a noising framework based on the second kind of noising (the most general, that is the one in which the variations of f are perturbed), but independent of the problem to solve and of the probability distribution of the noises, and with only one parameter: the CPU time fixed by the user.

Broadly speaking, this automatic version is made of a series of noising methods. The first one is very short. Then, the CPU time of each noising method of the series is the double of the CPU time of the previous noising method (thus, the last noising method of the series gets about half the total CPU time chosen by the user). The objective of each noising method of the series is double: of course to look for a good solution, but also to refine the tuning of the parameters provided by the previous noising method of the series. Because of this and of the doubling of CPU time, we may notice from the experiments ([17–19]) that this automatic tuning provide very good values for the parameters, at least as good as the ones that we can find by a fine manual tuning.

It has been tested out on four problems in [17–19] with two probability distributions (a uniform law and a logarithmic one). In the eight cases, the parameter-values and the solutions provided by the automatic noising method can be considered as very satisfying. For example, it is tested out in [17] on 5790 instances of the linear ordering problem (see Section 3.2): the automatic noising method found an optimal solution every time but six, which have been solved by a second application.

2.7. Variants

Independently of the different schemes that we can get by choosing the combination of structural or numerical parameters, it is possible to design some other variants, not depending on the noising methods in the sense that they can be applied to other metaheuristics. We detail two of them below.

- The first one consists in alternating noised phases with "unnoised" descents. More precisely, in order to stay closer to the genuine function, we may apply a given number NT of noised trials, then a descent with respect to the genuine function (i.e., with a 0-noise) until a local minimum is reached, then again NT noised trials, then, a descent with respect to the genuine function, and so on. For the first or third types of the noising method (a noise is added to the data or a part of the data is forgotten), we may alternate a noised descent (i.e., a descent with respect to the noised data; of course, new noises are computed before applying this noised descent) and an unnoised one (i.e., a descent with respect to the genuine data). For the second type (a noise is added to the variations of f), it appears from our experiments that a descent needs about $4 \cdot NS$ trials (where NS is the size of the neighbourhood); then, we apply successively $NT = 4 \cdot NS$ noised trials, an unnoised descent, and so on... For both cases, the process runs as if the noise-rate followed the behaviour depicted by Fig. 2 in which the length of all the stages (the number of trials for a fixed noise-rate) is about the same. This variant allows to check a good number of local minima (with respect to the genuine data) which could provide good solutions.
- The second variant consists in coming back to the best computed solution periodically. Indeed, because of the bad transformations randomly accepted, it may happen that we leave an interesting part of the space of solutions for a less interesting one. So one possible strategy is to periodically restart the current solution with the best solution found since the beginning (see Fig. 3). Of course, it

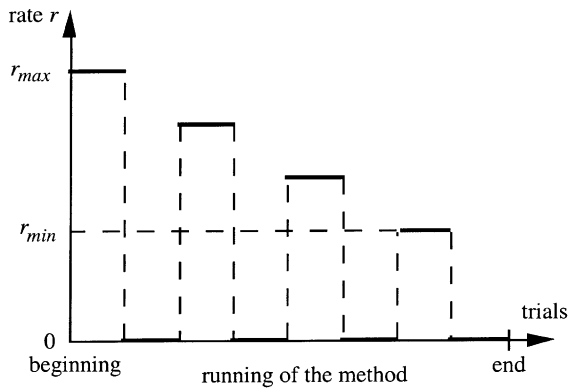

Fig. 2. Evolution of the noise-rate in the first variant.

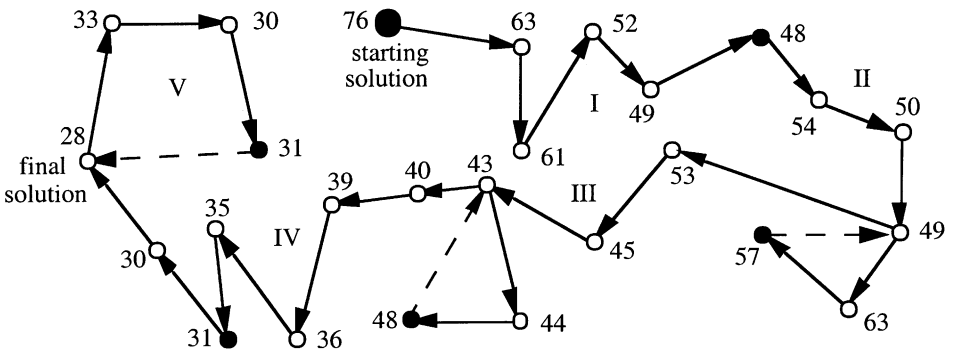

Fig. 3. Periodical restarts of the current solution with five restarts and $5 \cdot NS$ trials between two restarts. The values besides the circles give the value of f for the solutions associated with the circle. An arrow corresponds to NS trials; the white circles correspond to the current solution after NS trials. The little black circles show the restart-moments; the dashed lines correspond to the restart jumps, if any.

is not useful to restart the current solution too often. In order not to introduce a new parameter, the restart period was fixed as follows in [13]. Let $\alpha^2 \cdot NS$ be the total number of trials performed by the method (where NS is the neighbourhood-size; notice that α is not necessarily an integer); then, the current solution was restarted with the best computed solution after every cluster of about $\alpha \cdot NS$ trials (in other words, there are about α restarts), what seemed to give a good frequency (see also [16] for a study of the impact of the restart period).

By combining all these components, we may get many different schemes of the noising methods (including the simulated annealing method or the threshold accepting algorithm, as mentioned above). To give an example, we depict one of these schemes in the Appendix A, which is the one that we get by applying the second type of the noising methods (a noise is added to the variations of f), with a uniform noise drawn in $[-r, +r]$ where r decreases arithmetically from $r_{\max}$ down to $r_{\min}$ and with the two above variants. Another scheme may be found in [11].

3. Applications of the noising methods

As said above, the noising methods have been applied to different combinatorial problems arising in the following fields:

- the partitioning of a weighted graph into cliques: [11,15,18,19,27,28,42];
- the travelling salesman problem: [2,13,16, 19,30,31];
- the 0–1 multidimensional knapsack problem: [9,25,29];
- the linear ordering problem: [14,17,19];
- multi-criteria decision aid: [12];
- the covering problem and the packing problem in coding theory: [7,20,21];
- the alignment of graphemes and phonemes in linguistics: [43];
- the multi-resource generalized assignment problem: [3,6];
- the task allocation problem: [22];
- the prize-collecting Steiner tree problem: [10];
- the design of discrete manufacturing processes: [33];
- scheduling problems: [4,35,37].

In order to show how the noising methods can be applied from a practical point of view, we detail four applications below.

3.1. Graph partitioning

The first results ([11]) obtained by a noising method dealt with a NP-hard problem arising for instance in clustering and classification: given a complete non-oriented graph $G = (X, E)$ weighted by positive or negative integers, find a partition of

X into subsets (of which the number is not fixed) so that the sum of the weights of edges with their two extremities in the same subset is minimum.

In [11], we applied the first kind of the noising method: a noise is randomly added to the weights of the edges, with a uniform distribution into $[-r,+r]$ and an arithmetical decrease of the noise-rate r. We alternate noised descents with unnoised ones. There is a periodical restart of the current solution. The graphs studied in [11] have up to 150 vertices; the weights of the edges are integers belonging to $\{-1,1\}$ or to $[-w_{\max},w_{\max}]$ with $w_{\max}$ equal to 100 or to 1000. The g.e.t. consists in transferring a vertex from the subset to which it currently belongs to another one, which can be empty if we want to create a new subset (if a solution is coded by associating with each vertex the number of the subset to which it belongs, then this g.e.t. is a substitution). The examination of the neighbourhood for the descents (the noised ones as well as the unnoised ones) is systematic. The best tuning of the parameters depends on the studied graphs: the maximum rate-noise $r_{\max}$ is chosen between $0.8\times w_{\max}$ and $0.95\times w_{\max}$ and the minimum rate-noise $r_{\min}$ is chosen between $0.15\times w_{\max}$ and $0.5\times w_{\max}$; anyway, $0.8\times w_{\max}$ for $r_{\max}$ and $0.4\times w_{\max}$ for $r_{\min}$ always gave good results. Of course, the total number of trials depends on the CPU time that the user wishes to spend in order to solve his/her problem. For the frequency of the restart, if α denotes the number of pairs (noised descent, unnoised descent) that we apply between two restarts and β the number of restarts (so the total number of noised descents as well as unnoised ones is equal to $\alpha.\beta$), then a good tuning of α and β was to choose α between β and 2β.

We compared this scheme with the result got by simulated annealing with a classic pattern. The exploration of the neighbourhood is a random one, with the same g.e.t. as for the noising method. The initial temperature is computed by the method proposed in [34] with an initial acceptance ratio of 50%. The decrease of temperature is geometric with a ratio of 0.925. The number of proposed e.t.s is proportional to n^2 (where n is the number of vertices) with a proportionality coefficient varying from 2 to 15. The number of temperature changes is fixed, varying from 11 to 15. Of course, these parameters were tuned in order to be the best possible with respect to the amount of CPU time that we wanted to spend to solve our problem.

These values for the noising method and for simulated annealing are chosen so that it is possible to give about the same amount of CPU time to both methods. The conclusions of the experiments are that, for a same given CPU time (from about 100 to 1400 seconds a SUN workstation), the noising method performed slightly better than simulated annealing: the average and the worst values of the objective function f over 20 trials for each graph were always lower (sometimes with a relatively great difference) for the noising method than for simulated annealing; for the best value, it is often the same if the CPU time is great enough, otherwise, the noising method gave better results. Another way to compare the efficiencies of these methods is to compare the CPU time necessary to find the same average value of f. Then, it appeared that simulated annealing was much longer than the noising method: to reach the same value of f, simulated annealing had to run 1.7 to 3.3 times longer than the noising method. This was partially due to the fact that the noising method can benefit from a systematic exploration of the neighbourhood (see [11]).

3.2. Linear ordering problem

Given a set Π of m linear orders $P_1,P_2,\ldots,P_m$ defined on the same set X of "candidates", the linear ordering problem consists in finding a linear order minimizing the remoteness $f_\Pi(\omega)$ from Π over the set $\{\omega\}$ of linear orders defined on X. To define this remoteness, it is usual to consider the symmetric difference distance δ defined by

$$\delta(R_1,R_2)=|\{(x,y)\in X^2:[(xR_1y)\text{ and not }(xR_2y)]\text{ or }[\text{ not }(xR_1y)\text{ and }(xR_2y)]\}|$$

for any binary relations R_1 and R_2 defined on X. This quantity measures the number of disagreements between R_1 and R_2. Then, we set, for any linear order ω defined on X:

$$f_{\Pi}(\omega) = \sum_{k=1}^{m} \delta(P_k, \omega).$$

This quantity $f_{\Pi}(\omega)$ measures the total number of disagreements between the linear order ω and the set Π of preferences. The linear ordering problem is NP-hard.

There are two noising methods studied in [14]. The first one (N1) is called the "basic noising method" and the second one (N2) the "annealed noising method" because the noise distribution (a logarithmic noise) follows the usual scheme of simulated annealing. Some features are common to N1 and N2. First, the same shifting g.e.t. is applied; such a g.e.t. depends on two parameters and the size of the induced neighbourhood is $NS = |X| . (|X| - 1)$. The exploration of the neighbourhood is semi-exhaustive: one parameter of the g.e.t. varies cyclically and, for any fixed value of it, the best value of the second one is looked for. Second, the noise is added to the variation of the function to minimize. Third, the number of applied e.t.s is proportional to the size of the neighbourhood and is equal to $3000 \cdot NS$. Last, there is no restart and unnoised descents are not alternated with noised ones.

The main differences between N1 and N2 relies on the way of managing the noise. In N1, the noise is uniform and the noise-rate decreases arithmetically from 40 down to 0, each decreasing occuring after NS e.t.s are tried. In N2, the noise is logarithmic and the noise-rate decreases 40 times geometrically with a decreasing rate equal to 0.925 and from an initial value equal to 80.

We applied these two noising methods to problems with up to $|X| = 240$ candidates. We compared N1 and N2 to other methods as simulated annealing, and we hybridized them with a genetic algorithm. From our experiments, it appears that N1 and N2 found good solutions much quicker than a classic simulated annealing (SA): SA required $40,000 \cdot NS$ trials to get results comparable with those provided by N1 or N2 with only $3000 \cdot NS$ trials . With the same amount of trials, N1 seemed to give slightly better results than N2. It is worth noticing that, for SA, N1 and N2, the hybridization of these methods with a genetic algorithm improves them, at least if enough resources (the number of trials) are given to them (see [14] for details).

We applied also the automatic tuning of Section 2.6 to this problem in [17] and in [19], together with a branch and bound method to find an exact solution in [17]. This self-tuned noising method is based on N1. As said above, the parameters do not depend on the user, but are constant or computed (for $r_{\max}$ and for the number of trials) from the features of the problem that the user wants to solve. The restart and descent variants are applied: a restart occurs after about $\alpha \cdot NS$ trials if the total number of trials is $\alpha^2 \cdot NS$ and a descent is applied after $20 \cdot NS$ trials. With this tuning, the noising method found an exact solution in more than 98% of the cases. More precisely, we applied it on 5790 instances with different characteristics and with a number of candidates varying from 15 to 100: in only six cases (that we solved exactly by a second application), it did not find an optimal solution at first time but an approximate solution very close to an optimal one. As the noising method is stochastic, applying it twice to problems with a size and features similar to the ones that we studied should give an exact solution almost surely.

3.3. Travelling salesman problem

The travelling salesman problem (TSP) is well-known: find a minimum-weighted Hamiltonian cycle of a complete weighted graph. In [13,16,19], we paid attention to two types of TSPs. In the first one, the weights of the edges are randomly generated; in the second one, we deal with Euclidean TSP's in which the vertices are (randomly or not) distributed in the plane and the valuation of the edge $\{i, j\}$ is the Euclidean distance between i and j. These two problems are NP-hard. It is usual to adopt Lin's 2-opt ([36]) as the g.e.t. for the TSP (this g.e.t. can be considered as an inversion). This g.e.t. becomes an e.t. by the choice of two vertices which are not adjacent in the current Hamiltonian cycle, and so it induces a neighbourhood with $NS = n(n-3)/2$, if n denotes the number of vertices. For the Euclidean TSP, we divide the plane into square regions so that each region contains some vertices (about 3 or 4) as in [8] and the

application of Lin's 2-opt is restricted to vertices belonging to a same region or to two adjacent regions; so NS is quite less than in the general case, what allows to save CPU time by reducing the number of tried e.t.s.

The study developed in [16] deals with different ways of tuning the parameters and with the effects of these tunings on the results that we get. This study contributes to understand better the effects of the different parameters and variants, and helped us also in designing the automatic version of the noising methods ([19]).

In [13], we studied different ways of mixing components of simulated annealing and of the noising methods. Thus, we designed 10 methods, including six variants of simulated annealing and three kinds of noising methods: N, NRD and RN. In N, the noise is randomly chosen with a uniform distribution and is added to the variations of the objective function, the exploration of the neighbourhood is systematic, the noise decreases arithmetically from $r_{\max}$ to $r_{\min}$ after NS trials. The method called NRD is like N, but with the restart variant and the descent one (unnoised descents are alternated with noised ones). The only difference between N and RN is that there is a random exploration of the neighbourhood in RN instead of the systematic one of N.

Among our experiments, we reported in [13] the results provided by N, NRD and RN applied 50 times to two TSP's with 100 vertices and integer weights randomly chosen in [1, 100] and to four problems coming from [41] with a number of vertices varying from 76 to 152; for these problems, we chose $r_{\max} = 10,\ r_{\min} = 5$; different values were tested for the number of trials, from $100 \cdot NS$ to $10000 \cdot NS$. We applied them also to two Euclidean TSP's: one on 1000 vertices randomly distributed in the unit square, with $r_{\max} = 0.04, r_{\min} = 0$ and $1000 \cdot NS$ trials, one on 2500 vertices regularly located on the crosses of a grid, with $r_{\max} = 0.03, r_{\min} = 0$ and still $1000 \cdot NS$ trials. The comparison of the ten methods applied to these TSP's and to others gives the following conclusions.

- The rule acceptance used in the noising methods seems to be more efficient than the one used in simulated annealing when the number of trials is rather small, but less efficient for greater values.
- The restart and descent variants improve the methods quite a lot, especially when applied together and involve faster computations.
- A systematic exploration of the neighbourhood brings a slight improvement while reducing the CPU time by two.
- The two best methods (in these experiments) were NRD and an "adaptable" simulated annealing in which the current temperature may increase or decrease, according to the number of e.t.s adopted between two temperature-changes (if this number is not high enough, the temperature increases).

3.4. Coding theory

The following problem arises from coding theory. Let q and n be two positive integers with $q \geqslant 2$. Let Z_q denote the set $\{0, 1, \ldots, q-1\}$ and let Z_q^n be the set of all n-tuples defined over Z_q. A set $C \subseteq Z_q^n$ of n-tuples is called a q-ary (n, M, d)-code if $| C | = M$ and if d is the smallest number of positions in which two distinct elements of C differ (this corresponds with the Hamming distance). One of the most basic problems of coding theory is to find the largest code of given length (n) and given minimum distance (d). The largest value of M such that there exists a q-ary (n, M, d)-code is denoted by $A_q(n, d)$. The values of $A_q(n, d)$ have been studied extensively (by simulated annealing, genetic algorithms, or other methods) and exact values or sharp bounds are known for many pairs (n, d).

To apply a noising method to the computation of $A_4(n, d)$, Bogdanova ([7]) defines the quantities $A_i (1 \leqslant i \leqslant d-1)$ for a set $C \subseteq Z_4^n$ with $| C | = M$ by:

$$A_i = \frac{1}{M} \mid \{(x, y) \in C^2 \text{ such that } d(x, y) = i\} \mid$$

then, she sets $f(C) = \sum_{i=1}^{d-1} A_i$. The aim is then to find a set C with $| C | = M$ and $f(C) = 0$. In this case, $A_4(n, d)$ is greater than or equal to M. Then, a new set C' is considered with $| C' | = M + 1$ and the previous process is applied again.

The scheme used in [7] follows the one proposed in [11]: the noise is added to the data, not to the variations of f. To do this, Bogdanova gives a random value $k(v) \in [1-r, 1+r]$ to each vector v belonging to Z_4^n, where r is the noise-rate; the distribution of k is uniform. The noised function f_{noised} is defined by (with the same notations as above):

$$f_{\text{noised}}(C) = \frac{1}{2M} \sum_{i=1}^{d-1} \sum_{(x,y)\in C^2, d(x,y)=i} [k(x) + k(y)].$$

Notice that, when the noise-rate r is equal to 0, then, $k(v) = 1$ for any v and so f and f_{noised} coincide. The g.e.t. is a substitution consisting in changing one position of one element of the current set C. The noise-rate decreases arithmetically from 1 down to 0. An unnoised descent is applied after each noised descent. By means of this noising method, Bogdanova succeeded to break several lower bounds for $q = 4$ (see [7] for details and for the new records).

4. Conclusions

It would be unwise to conclude that the noising methods constitute a panacea in the field of combinatorial optimization! If we detailed some successful applications above, we can also quote two experiments for which we failed to find better results than the ones reported in the literature. The first one deals with a problem of coding theory close to the one described in Section 3.4: how many disjoint spheres with a given radius is it possible to select in $\{0,1\}^n$ for a given dimension n (the ternary version of this problem is also known as "the football team" problem)? We got some of the known lower bounds but we did not succeed to break records (we got new ones in [21], but these new records appeared simultaneously). The second one deals with the computation of some Ramsey numbers: given two integers c and i, the Ramsey number $R(c,i)$ is the minimum integer so that any graph on $R(c,i)$ vertices contains at least a clique on c vertices or an independent set of i vertices. The computation of these numbers is difficult, even for small values of i and c. We applied a noising method to the computation of $R(c,i)$ for some values of c and i, but without succeeding to break records. Maybe these failures come from the fact that these problems are more "structural" than "numerical" ones.

Nonetheless, even if the noising methods do not constitute a panacea, we claim that they deserves interest because they provide, as well as (and sometimes better than) other methods like simulated annealing, a way of solving hard problems with a usually "good" solution within a "reasonable" CPU time. The examples developed above show that we can sometimes find better results and more quickly than a classic simulated annealing, especially if we try to make these methods collaborate by an appropriate hybridization rather than considering them as rivals. We hope that other researchers will try to apply the noising methods, alone or hybridized with other heuristics, to other problems; our aim will be reached if this review can stimulate them to do so and can help them in adapting the noising methods to their problems.

Appendix A

The following scheme of the noising method is the one that we get by applying the second type of the noising method (a noise is added to the variations of f), with a uniform noise drawn in $[-r, +r]$ where r decreases arithmetically from $r_{\max}$ down to $r_{\min}$, for some exploration of the neighbourhood and with the two variants described in Section 2.7: an unnoised descent is applied after $4 \cdot NS$ noised trials, where NS denotes the size of the neighbourhood, and we restart the current solution after $\alpha \cdot NS$ noised trials, if the total number of noised trials is $\alpha^2 \cdot NS$ (here, α is assumed to be an integer). Another scheme may be found in [11]. Comments are given between asterisked parentheses.

```
Parameters
  r_max, r_min: reals                (* give the initial and final noise-rates *)
  total_nb_trials: integer           (* gives the total number of noised elementary *)
                                     (* transformations that should be applied *)
  nb_trials_at_fixed_rate: integer   (* gives the number of noised elementary *)
                                     (* transformations that should be tried with a fixed noise-rate *)
Constant
  NS: integer                        (* NS is the size of the neighbourhood *)
                                     (* induced by the generic elementary transformation *)
Variables
  s, s' : of type "solution"         (* s is the current solution and s', one of its neighbours *)
  best_sol: of type "solution"       (* best_sol is the best solution found since the beginning *)
  trial_meter: integer               (* counts the number of "noised" trials which have been applied *)
  rate: real                         (* gives the current value of the noise-rate *)
  noise: real                        (* gives the value of a noise to be added to the variation of f *)
  decrease: real                     (* denotes the value by which rate decrease *)
  restart: integer                   (* gives the frequency of the restart of the current solution *)
Begin
  draw the initial current solution s randomly
  best_sol ← s
  decrease ← (r_max − r_min)/[(total_nb_trials/nb_trials_at_fixed_rate) − 1]
  restart ← sqrt(total_nb_trials × NS)
  rate ← r_max
  trial_meter ← 0
  while trial_meter < total_nb_trials, do
    begin
      trial_meter ← trial_meter + 1
      let s' be the next neighbour of s   (* "next" with respect to the adopted *)
                                          (* exploration of the neighbourhood *)
      let noise be a random real number uniformly drawn into [−rate, rate]
      if f(s') − f(s) + noise < 0, then s ← s'
      if trial_meter = 0 (modulo 4 · NS), then apply an "unnoised" descent from s
      if f(s) < f(best_sol), then best_sol ← s
      if trial_meter = 0 (modulo restart), then s ← best_sol
      if trial_meter = 0 (modulo nb_trials_at_fixed_rate), then rate ← rate − decrease
    end of "while"
end.
```

References

[1] E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, New York, 1997.

[2] B. Alidaee, Experimental Design for Combinatorial Optimization Problems, INFORMS, 1995, New Orleans, US, 1995.

[3] B. Alidaee, M.M. Amini, G.A. Kochenberger, Hybrid metaheuristic approaches to the multi-resource generalized assignment problem: A case study of tabu search, space smoothing, and noising methods, Second Metaheuristics International Conference (MIC-97), Sophia-Antipolis, France, 1997.

[4] B. Alidaee, J.T. Naidu, E.L. Gillenwater, Heuristic algorithms for multiple scheduling with the objective of minimizing total weighted and unweighted tardiness, INFORMS 1997, Dallas, TX, 1997.

[5] I. Althöfer, K.-U. Koschnick, On the convergence of 'threshold accepting', Applied Mathematics and Optimization 24 (1991) 183–195.

[6] M.M. Amini, B. Alidaee, M.J. Racer, Alternative metaheuristics approaches to the multi-resource generalized assignment problem, INFORMS 1997, Dallas, TX, 1997.
[7] G. Bogdanova, Optimal codes over an alphabet of four elements, in: The Proceedings of the Fifth International Workshop on Algebraic and Combinatorial Coding Theory, Unicorn, Shumen, Sozopol, Bulgaria, 1996, pp. 46–53.
[8] E. Bonomi, J.-L. Lutton, The *N*-city travelling salesman problem: Statistical mechanics and the Metropolis algorithm, SIAM Review 26 (1984) 551–568.
[9] P. Boucher, G. Plateau, Étude des méthodes de bruitage appliquées au problème du sac à dos à plusieurs contraintes en variables 0–1, in: The Proceedings of the Cinquièmes journées nationales sur la résolution pratique de problèmes NP-complets (JNPC '99), Lyon, France, 1999, pp. 151–162.
[10] S.A. Canuto, C.C. Ribeiro, M.G.C. Resende, Local search with perturbations for the prize-collecting Steiner tree problem, in: The Proceedings of the Third Metaheuristics International Conference (MIC-99), Angra dos Reis, Brazil, 1999, pp. 115–119.
[11] I. Charon, O. Hudry, The noising method: A new combinatorial optimization method, Operations Research Letters 14 (1993) 133–137.
[12] I. Charon, O. Hudry, An application of the noising method to MCDA, Invited communication to "EURO XIV", Jérusalem, Israël, July 1995.
[13] I. Charon, O. Hudry, Mixing different components of metaheuristics, in: I.H. Osman, J.P. Kelly (Eds.), in: Metaheuristics: Theory and Applications, Kluwer, Boston, 1996, pp. 589–603.
[14] I. Charon, O. Hudry, Lamarckian genetic algorithms applied to the aggregation of preferences, Annals of Operations Research 80 (1998) 281–297.
[15] I. Charon, O. Hudry, Variations on the noising schemes for a clustering problem, in: The Proceedings of the Third Metaheuristics International Conference (MIC-99), Angra dos Reis, Brazil, 1999, pp. 147–150.
[16] I. Charon, O. Hudry, Application of the noising method to the travelling salesman problem, European Journal of Operational Research 125/2 (2000) 266–277.
[17] I. Charon, O. Hudry, A branch and bound algorithm to solve the linear ordering problem for weighted tournaments, Discrete Applied Mathematics, to appear.
[18] I. Charon, O. Hudry, Noising methods for a clique partitioning problem, submitted for publication.
[19] I. Charon, O. Hudry, Automatic tuning of the noising methods, submitted for publication.
[20] I. Charon, O. Hudry, A. Lobstein, A new method for constructing codes, in: The Proceedings of the Fourth International Workshop on Algebraic and Combinatorial Coding Theory, Novgorod, Russie, 1994, pp. 62–65.
[21] I. Charon, O. Hudry, A. Lobstein, Application of the Noising Method to Coding Theory, IFORS 96, Vancouver, Canada, 1996.
[22] W.-H. Chen, C.-S. Lin, A hybrid heuristic to solve a task allocation problem, Computers and Operations Research 27 (2000) 287–303.
[23] G. Dueck, T. Scheurer, Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing, Journal of Computational Physics 90 (1990) 161–175.
[24] G. Dueck, J. Wirsching, Threshold accepting algorithms for multi-constraint 0–1 knapsack problems, Technical paper, TR 89 10 016, IBM Heidelberg Scientific Center, Germany, 1989.
[25] A. Fréville, S. Hanafi, A. El Abdellaoui, Noising method for the 0–1 multidimensional knapsack problem, FRANCORO, Mons, Belgique, 1995.
[26] F. Glover, M. Laguna, Tabu Search, Kluwer, Dordrecht, MA, 1997.
[27] D. Guillaume, F. Murtagh, An application of XML and XLink using a graph-partitioning method and a density map for information retrieval and knowledge discovery, in: D.M. Mehringer, R.L. Plante, D.A. Roberts (Eds.), Astronomical Data Analysis Software and Systems VIII, ASP Conference Series 172, ASP, San Francisco, 1999, pp. 278–282.
[28] D. Guillaume, F. Murtagh, Clustering of XML documents, in: The Proceedings of From Information to Knowledge using Astronomical Databases, Computer Physics Communications, to appear.
[29] S. Hanafi, A. Fréville, A. El Abdellaoui, Comparison of heuristics for the 0–1 multidimensional knapsack problem, in: I.H. Osman, J.P. Kelly (Eds.), in: Metaheuristics: Theory and Applications, Kluwer, Boston, 1996, pp. 449–465.
[30] C.-P. Hwang, Global and local search heuristics for the symmetric travelling salesman problems, Ph.D. thesis, University of Mississippi, 1996.
[31] C.-P. Hwang, B. Alidaee, J.D. Johnson, A tour construction heuristic for the travelling salesman problem, Journal of the Operational Research Society 50 (1999) 797–809.
[32] S.H. Jacobson, K.A. Sullivan, New convergence results of generalized hill climbing algorithms, INFORMS 1998, Montreal, Canada, 1998.
[33] S.H. Jacobson, K.A. Sullivan, A.W. Johnson, Discrete manufacturing process design optimization using computer simulation and generalized hill climbing algorithms, Engineering Optimization 31 (1998) 247–260.
[34] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation – Part II (graph coloring and number partitioning), Operations Research 39 (3) (1991) 378–406.
[35] R.B. Kethley, Single and multiple machine scheduling to minimize total weighted late work: An empirical comparison of scheduling rules, algorithms and meta-heuristics using Taguchi loss functions, Ph.D. thesis, University of Mississippi, US, 1997.
[36] S. Lin, Computer solutions for the travelling salesman problem, Bell System and Technology Journal 44 (1965) 2245–2269.

[37] J. Naidu, B. Alidaee, E. Gillenwater, Heuristic approaches to minimize tardiness problems on single machine with sequence dependent set-up times, INFORMS 1997, Dallas, US, 1997.
[38] I.H. Osman, J.P. Kelly, Metaheuristics: An overview, in: I.H. Osman, J.P. Kelly (Eds.), in: Metaheuristics: Theory and Applications, Kluwer, Boston, 1996, pp. 1–21.
[39] M. Pirlot, General local search methods, European Journal of Operational Research 92 (1996) 493–511.
[40] C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, London, 1995.
[41] G. Reinelt, TSPLIB – A traveling salesman problem library, ORSA Journal on Computing 3 (4) (1991) 376–384.
[42] V. Sudhakar, C. Siva Ram Murthy, A modified algorithm for the graph partitioning problem, Integration, the VLSI Journal 22 (1997) 101–113.
[43] F. Yvon, Prononcer par analogie: Motivation, formalisation et évaluation, Ph.D. thesis, ÉNST, Paris, 1996.