The noising methods: a survey

I. Charon and O. Hudry

École nationale supérieure des télécommunications
46, rue Barrault, 75634 Paris cedex 13, France
Irene.Charon@enst.fr, Olivier.Hudry@enst.fr

Abstract: The aim of this paper is to summarize the principles and the applications of the *noising methods*, a recent family of combinatorial optimization metaheuristics. We describe their commons features and their variants and we give the list of their applications to different combinatorial optimization problems. We also show how the simulated annealing algorithm and the threshold accepting algorithm can be considered as noising methods when the components of the noising methods are properly chosen.

1 The origins of the noising methods

At the beginning of the 1990's, we were studying the application of genetic algorithms to combinatorial optimization problems with strong structural constraints, like the ones depicted below. Unfortunately, we got only poor results, even when compared to the ones provided by repeated descents. So, we thought that maybe the genetic algorithms would be more appropriate if the environment, measured in some ways by the function f that we want to optimize, could fluctuate, as it happens in real life with genetics.

Thus we considered that f is the limit of an evolving function, or equivalently that we do not know precisely the values taken by f: these values are perturbed by *noises*; the range of the noises decreases to zero so that the evolving function converges towards f. Of course, to get an idea of the new performances of our genetic algorithms, we applied the same evolving process to the repeated descents. While the results got by the genetic algorithms remained as poor as before in our experiments, the surprise came from the repeated descents: the results that they provided at the end of this evolving process were much better than the ones got previously with the same CPU time.

The principle of the noising methods was born: to perturb the values taken by f by decreasing noises while applying a local search method. Then we designed in 1992 the first version of a noising method and this one appeared in 1993 [10]. Since this date, we have been studying different ways to perturb f and different variants of what has become a family of metaheuristics that we still call *noising methods* because they share the same basic principles.

As for the other metaheuristics (for references on metaheuristics, see for instance [1, 36, 37, 39]), the noising methods are not designed to solve only one special

type of problems, but to be applicable to various kinds of combinatorial optimization problems. Such a problem may be described as follows: given a finite set S and a function f defined on S, find the optimum (the maximum or the minimum) of f over S and an element of S optimizing f. As maximizing a function f is the same as minimizing $-f$, we may assume without loss of generality that we deal with minimization problems. Thus we shall consider from now on a problem with the following structure:

$$\text{Minimize } f(s) \text{ for } s \in S.$$

The aim of this paper is to present the principles and the applications of the noising methods. We detail their commons features and their variants and we list their applications to different combinatorial optimization problems. We also study the links between them and two other metaheuritics: the simulated annealing algorithm and the threshold accepting algorithm; we show how these two metaheuristics can be considered as noising methods when the components of the noising methods are properly chosen.

The paper is organized as follows: the following section recalls some basic considerations about the notion of *neighborhood*. Then, in section 3, we describe the main principles of the noising methods. Section 4 is devoted to the links between the noising methods and simulated annealing or threshold accepting methods. The applications of the noising methods are listed in Section 5, just before the conclusions.

2 Elementary transformations, neighborhoods and descents

Many metaheuristics (sometimes called also *local search methods*) applied to combinatorial optimization problems are based on *elementary* (or *local*) *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 could be to change, for a given i, the bit located in position i in s into its complement. For a given elementary transformation scheme, the set of the solutions s' that we can get by applying such an elementary transformation to a given solution s is called the *neighborhood* $N(s)$ of s, and the elements s' of $N(s)$ are called the *neighbors* of s.

Thanks to such an elementary transformation, we can design an *iterative improvement method* (also called a *descent* for a minimization problem): from the current solution s, we consider a neighbor $s' \in N(s)$ of s; if we have $\Delta f(s,s') < 0$ with $\Delta f(s,s') = f(s') - f(s)$, then s' becomes the new current solution, otherwise we keep s as the current solution and we try another neighbor of s. Then, we do

it again with the current solution until it is impossible to find a neighbor of the current solution s^* which is better than s^*:

$$\forall s' \in N(s^*), f(s') \geq f(s^*).$$

Thus, the solution s^* provided by a descent is a local minimum with respect to the adopted elementary transformation (or, similarly, with respect to the neighborhood induced by it) and is not necessarily an optimal solution.

There exist several ways to explore the neighborhood of a solution s. We recall three of them below. It is of course possible to imagine some other ways to explore the neighborhood, including by mixing the three following strategies.

- It can be done *at random* (as in a classic simulated annealing): then s' is randomly chosen in $N(s)$. One drawback of this strategy is that it is memoryless. We may scan the same neighbor several times, instead of considering another neighbor. Moreover, two successively generated neighbors are globally more different than when generated systematically (see below); then it is more difficult to benefit by the previous trials in order to reduce the amortized complexity, what usually involves a great amount of CPU time. Finally, 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 more efficiently than the strategies below. Similarly, if the size of the neighborhood is very large, it can be the best way of looking for an interesting neighbor.

- The exploration can also be *systematic* or *cyclic* (see [23]): the neighbors are ranked in a certain order and they are all considered in this order once, before being considered for a second time and a neighbor better than the current solution is accepted as soon as it has been discovered; for a descent, this process is applied until a complete cyclic exploration of the neighborhood has been performed without finding any better neighbor: then the final solution is a local minimum with respect to the adopted elementary transformation.

- The third classic exploration, that we can call *exhaustive*, is the one applied for instance in a classic tabu search or in a deepest descent: the whole neighborhood is explored in order to find the best neighbor of the current solution. The main drawback of this strategy is that, when the size of the neighborhood is quite large, it can be very long to explore it entirely between two adopted transformations.

The noising methods are based on elementary transformations. The next section is devoted to the principles and variants of these methods.

3 Principles of the noising methods

As said above, to minimize the function f, the noising methods do not take the true values of f into account but consider that they are perturbed in some way by *noises*

in order to get a "noised" function f_{noised}. It means that, with respect to what happens in a descent, when the variation $\Delta f(s, s')$ is computed in order to know whether s' is accepted instead of s or not, we do not compute $\Delta f(s, s')$ from the true values taken by f, but we consider that these values taken by f are perturbed by *noises* and thus we compute in fact the variation $\Delta f_{noised}(s, s')$. During the run of the algorithm, the range of the perturbing noises decreases (typically to zero, but it often happens that we may stop the process before), so that, at the end, there is no significant noise and the optimization of f_{noised} leads to the same solution as the one provided by a descent applied to f with the same initial solution. Thus it is necessary to specify how to perturb f to get a noising method.

More generally, several questions must be answered in order to design a noising method scheme:

1. How to perturb f in order to get f_{noised}?
2. How and when to make the range of noise decrease?
3. How to choose the maximum (and initial) and the minimum (and final) values of the range of noise?
4. How to explore the neighborhood?
5. Is it possible to add other ingredients to get interesting variants?

Of course, the answers to these questions depend on the problem to solve and on what the user wants to do: he or she has the possibility to choose his or her own components, and so, by designing his or her own noising method scheme, to diversify his or her investigations. Nonetheless, we give below some possible answers to the previous questions.

3.1 How to perturb f?

Though this distinction is a little contrived, we may distinguish three main ways to perturb f: by *noising the data*, by *noising the variations of* f, or by *forgetting a part of the data*. They are described more precisely below.

3.1.1 Noising the data

The first possibility to perturb f consists in adding the noises to the data. For instance, if f is a linear function of n variables x_i (with $1 \leq i \leq n$):

$$f(s) = \sum_{i=1}^{n} a_i x_i,$$

we may perturb the data a_i by adding some noises r_i to them:

$$f_{noised}(s) = \sum_{i=1}^{n} (a_i + r_i) x_i.$$

We may insert this way of noising the data in the following noising method scheme:

- Initializations
- repeat
    add noises to the data to get f_{noised}
    apply a descent to f_{noised} from the solution computed
        at the previous iteration
    reduce the range of noise
- until the range of noise is low enough
- apply a final descent to f from the solution provided by the previous loop.

In other words, we first compute an initial solution. Then, for each repeat-loop, noises are added to the data and a descent is applied with respect to these noised data, i.e., to f_{noised}; the solution found during this application of the descent becomes the initial solution of the next application of the descent. The range of noise decreases between two descents to a fixed value (for example zero). Notice that it is not always necessary to apply a complete descent to f_{noised}: for instance, the exploration of the neighborhood of the local (with respect to the noised data) minimum found by the descent is systematically useless. Thus it is often better (because it saves CPU time that we can use differently) to apply just the beginning of a descent and to stop it when a given number of (adopted or not) elementary transformations has been performed (for instance, the number of elementary transformations necessary to scan a neighborhood a few times). Anyway, it is better to apply the final descent to f completely.

Usually, the added noises are chosen randomly according to a given law of probability. For instance, in the first design of a noising method [10], the noises r_i $(1 \leq i \leq n)$ are chosen with a uniform law into an interval $[-r, +r]$, where r is the rate of noise at the considered iteration, and the initial (and maximum) value r_{max} of r depends on the data. But, here also, we may imagine different possibilities to choose the noises. Some examples are:

- a_i becomes $a_i + r_i$;
- a_i becomes $a_i \times (1 + r_i)$;
- the probability law is not necessarily uniform; it can be a Gaussian law or something else;
- the interval in which r_i is drawn is not necessarily centered on 0.

This pattern has been applied in particular in the first noising method [10] and, since, in [2-4, 6-9, 11, 20-22, 26-32, 34, 35, 38].

3.1.2 Noising the variations of f

We tried another possibility (see [12]): it consists in perturbing the variations of f. When a neighbor s' of s is tried, we do not consider the genuine variation $\Delta f(s,s')$ but a noised variation $\Delta f_{noised}(s,s')$ defined by:

$$\Delta f_{noised}(s,s') = \Delta f(s,s') + r(s,s')$$

where $r(s,s')$ denotes the noise (in fact, it depends not only on s and s', but also on the number of the iteration: if s and s' are considered twice, the noise $r(s,s')$ is not necessarily the same).

As for before, the law of probability followed by the noises $r(s,s')$ can be uniform on an interval centered at the origin or not, or Gaussian, or something else (for instance, we tried a distribution involving the hyperbolic tangent of the variation of f in the first version of [12]). This way of perturbing f by adding the noises to its variations has been developped in [12-18] and [40] (similar ideas are also considered in [33] to study the convergence of such metaheuristics). We will see latter that an appropriate choice of the law of probability (and of the other elements of the noising method scheme) leads to the pattern of a classic simulated annealing or of a threshold accepting algorithm.

The pattern of this kind of noising can be the following:

- Initializations
- repeat
    repeat
        consider a neighbor s' of the current solution s
        compute the noised variation of f:
            $\Delta f_{noised}(s,s') = \Delta f(s,s') + r(s,s')$
            where $r(s,s')$ denotes the random noise
        if $\Delta f_{noised}(s,s') < 0$, then adopt s' instead of s
    until a first criterion is fulfilled
    reduce the range of noise
- until a second criterion is fulfilled
- apply a final descent to f from the solution provided by the previous loop.

In this pattern, the first criterion (in the inner repeat-loop) gives the number of elementary transformations tried with a given rate of noise. The second criterion gives the number of decreasings of the rate of noise.

3.1.3 Forgetting a part of the data

The third and most recent way [14] (similar ideas are developped in [9]) consists in perturbing f by forgetting a part of the data. For instance, in [15], we consider the problem consisting in partitioning the vertex-set of a graph of which the edges have negative or positive weights in order to minimize the sum of the weights of the edges

with their two extremities in a same set. Then, we designed two ways to "forget" a part of the data:

- in the first one, a part of the vertices is selected and the weight of an edge is taken into account for the computation of f_{noised} only if its two extremities have been selected; in this process, the rate of noise (less than or equal to one) 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} if and only if it has been selected (for the problem tackled in [15], it is the same as giving a weight equal to zero to a forgotten edge).

The scheme of such a noising method can be the following:

- Initializations
- repeat
    forget a proportion of data equal to the rate of noise
    apply a descent to the remaining data
        from the solution computed at the previous iteration
    reduce the range of noise
- until the range of noise is low enough
- apply a final descent to f from the solution provided by the previous loop.

As for the noising of the data, it is not always necessary to apply a complete descent (except the last one).

This way of noising f gave very good results for the functions studied in [9] and in [15], but it is not obvious that it would be the general case: more experiments are still necessary to draw conclusions.

It is worth noticing that, from a theoretical point of view, the first and third ways to noise f are in fact special cases of the second type, since the rule to known whether a neighbor s' is accepted or not instead of the current solution s comes down to know whether a number depending on s and s' (the noised variation of f) is negative or not. For instance, for the traveling salesman problem with the neighborhood induced by the usual 2-opt transformation, noising the data with a uniform noise is the same as noising the variation with a noise following a probability law given by the sum of four uniform laws. Anyway, it is usually more convenient to apply the first or the third ways of noising f than their equivalents with respect to the noising of the variations, because it can be less natural, more tedious, and sometimes very difficult to find the equivalent probability law.

3.2 How and when to make the range of noise decrease?

In the first noising method, the rate of noise r decreases arithmetically during the running of the algorithm from a maximum rate of noise value r_{max} to zero (thus the arithmetical decreasing ratio, the maximum rate of noise and the total number of applied noised descents are related). It is also the case for the majority of the applications reported in this paper, sometimes with a minimum value r_{min} of the rate of noise which is not equal to zero. Then, if N denotes the total number of decreasings, the arithmetical decreasing rate is equal to $(r_{max} - r_{min})/N$.

Nevertheless, other kinds of decreasing have been studied. More precisely, a geometrical decreasing was tried in [12, 13], associated with a "logarithmic noise" (the noise follows a probability law given by the logarithm of a uniform law) as well as an arithmetical one associated with a uniform noise. Still in [12], we tried another possibility (for a noising method scheme inspired by simulated annealing) in which the rate of noise was allowed to increase from time to time. More precisely, let k be a given number; a function α is designed to give the number $\alpha(i)$ of bad elementary transformations that we would like to accept between the iterations i and $i + k$; if the number of bad elementary transformations really accepted between iterations i and $i + k$ is less than $\alpha(i)$, then the rate of noise increases, otherwise it decreases. The experiments done in [12] for the traveling salesman problem show that this "adaptive" method can lead to very good results, often better than the ones provided by the same method but with a decreasing rate of noise.

It seems that the frequency of the decreasing is not a crucial point, at least of course if its value is not too low (for instance, decreasing the rate of noise from its maximum value r_{max} to zero suddenly is usually not a good choice). For example, the rate of noise may decrease after each elementary transformation, at least theoretically; this could avoid to deal with an extra parameter to tune. Anyway, for practical reasons, it can be more convenient to make the rate of noise decrease less often, for instance when an exploration of the neighborhood is completed (or when a number of elementary transformations equal to the size of the neighborhood has been performed).

3.3 How to choose r_{max} and r_{min}?

3.3.1 Choice of r_{max}

The choice of r_{max} seems to be the most important one (of course after the choice of the number of elementary transformations that the user wants to apply to solve his or her problem; this value is directly related to the CPU time that he or she accepts to spend). So, in general, it is a paramater that the user must tune.

Anyway, some experiments done with the traveling salesman problem (see [16]) show that the sensitiveness of this parameter is not very sharp: a value of more or less 10 % from the "best" value of r_{max} gives almost the same final result and even a value of more or less 20 % gives very good results. One consequence is that it is

not always necessary to find a sharp tuning for r_{max}.

Moreover, it is sometimes possible to tune r_{max} automatically, as in [18] (see also [15] and [17]). In this paper, we designed a generic automatic noising scheme which can be applied with various types of noise or to various problems by changing only the type of noise or the features defining the problem; no parameter of this generic method is changed from one noising method scheme to another or from one problem to another. The only parameter given by the user is the CPU time that he or she wishes to spend to solve his or her problem. Broadly speaking, the principle of this automatically tuned variant consists in applying the considered noising method several times, in the sense that the rate of noise decreases several times from a maximum value to zero. The duration of each run is twice the one of the run performed just before; the first run is very short. This repetition of the noising method succeeded, for the problems in [15, 17, 18], in computing a suitable value for r_{max} which is improved during the process (see [18] for details).

3.3.2 Choice of r_{min}

In general, r_{min} is a parameter to tune. It must be chosen so that minimizing f_{noised} with a rate of noise less than r_{min} would lead to the same result as minimizing f itself. If so, it is clearly useless to try such values for the rate of noise: we can only waste CPU time without improving the current solution. So, choosing a value not equal to zero for r_{min} may save CPU time. Anyway, if the user does not want to tune an extra parameter, it is always possible to choose $r_{min} = 0$, even if it consumes a little more CPU time. When it is not easy to have a broad idea of a good value for r_{min}, it can even be a good deal to do so, because tuning a parameter is not always obvious, especially if the user is not an expert. Moreover, a too high value of r_{min} may damage the efficiency of the method very much. So, in the automatically tuned noising method of [18], we preferred to fix $r_{min} = 0$ and to increase the CPU time a little bit, rather than having to tune r_{min}.

3.4 Exploration of the neigborhood

In the majority of the applications of the noising methods reported here, the exploration of the neighborhood is systematic (or cyclic; see Section 2). But it happens that the exploration is a random one (especially when the noises perturb the data, as described in Section 3.1.1), or even partly systematic and partly random. For instance, if the elementary transformation involves two parameters (as for example the 2-opt transformation usually applied to the traveling salesman problem), the exploration can be done randomly on the first parameter and, for each value of the first parameter, the exploration can be systematic.

As said above, the main advantages of a systematic exploration are that we may sometimes benefit by the scanning of the previous neighbors to reduce the amortized complexity and to increase the diversity of the exploration by avoiding

to scan twice the same neighbor before scanning a neighbor not yet considered. Another advantage of such a strategy is that we can save CPU time with respect to an exhaustive exploration, but also with respect to a random one because drawing random numbers is a procedure which can consume a rather great amount of CPU time.

3.5 Other ingredients

Independently of the different schemes that we can get by combining the possibilities described above, it is possible to design some other variants. The two ones that we detail below can be applied to other metaheuristics. In our experiments with noising methods, they often appear fruitful.

3.5.1 Alternation of noised and unnoised phases

The first variant consists in alternating noised phases with unnoised descents. More precisely, in order to stay closer to the original function f, we may apply a given number ν of elementary transformations with respect to the noised function f_{noised}, then a descent with respect to f until a local minimum is reached, then again ν noised trials, then a descent with respect to f, and so on. This variant usually allows to check a fair number of local minima (with respect to the original data) which could provide good solutions.

For the first type of noising methods (noises are added to the data), we may alternate a noised descent (that is, a descent with respect to f_{noised}; of course, new noises are computed before applying each noised descent) and an unnoised one (that is, a descent with respect to f). For the second type (noises are added to the variations of f), it seems to be a good choice to give to the noised phase a number of elementary transformations which is about the same as the number of elementary transformations performed by a descent. From our experiments, it appears that a descent performs an average of about 4σ elementary transformations, where σ is the size of the neighborhood. Then we can perform successively $\nu = 4\sigma$ noised elementary transformations, an unnoised descent, ν noised elementary transformations, an unnoised descent, and so on.

3.5.2 Periodic restarts from the best computed solution

The second variant consists in coming back to the best computed solution periodically. Indeed, because of the bad transformations sometimes 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. Of course, it is useless (and even harmful) to restart the current solution too often. In order not to introduce a new parameter, the restart period was fixed as follows in [12]. Let $\gamma^2\sigma$ be the total number of elementary transformations performed by the method (where σ is the size of the neighborhood);

then the current solution is restarted with the best computed solution after every cluster of about $\gamma\sigma$ elementary transformations (in other words, there are about γ restarts), what seemed to give a good frequency.

4 The noising methods as generalizations of other metaheuristics

By combining all the ingredients detailed above, we may get many different noising method schemes, including the simulated annealing method or the threshold accepting algorithm. It is what we show now (the possibility of such a generalization is also noticed in [33]).

4.1 Links between the noising methods and simulated annealing

Indeed, we may consider that the second type of noising (noises are added directly to the variations of f) is a generalization of simulated annealing if we choose properly the parameters, especially the probability distribution.

In simulated annealing, the current solution s is replaced by one of its neighbors 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:

$$\Delta f(s,s') + \theta \ln p < 0.$$

Thus, it is the same result as adding to the variation of f a noise equal to $\theta \ln p$. Here the probability law is given by the logarithm of a uniform random variable drawn into]0,1[, and θ gives the value of the rate of noise. Then it is easy to choose the other ingredients of the noising method scheme to get a classic simulated annealing (in particular, the decreasing of the rate of noise θ will be geometric in this scheme; the exploration of the neighborhood is usually a random one for simulated annealing, though a systematic one is sometimes applied, see [23]). Variants of this "logarithmic scheme" have been studied in [12, 13, 15], as well as a uniform distribution.

4.2 Links between the noising methods and threshold accepting algorithms

Similarly, we may consider that the noising methods are a generalization of the threshold accepting algorithms designed by Dueck, Scheurer and Wirsching [24, 25].

In such a method, the current solution s is replaced by one of its neighbors s' if the variation $\Delta f(s,s')$ does not get bigger than a given threshold. This threshold depends on the iteration and decreases during the process to zero. So, with respect to simulated annealing, the main difference relies in the fact that the acceptance criterion is no longer the exponential Metropolis criterion. More precisely, if k denotes the current iteration, s' is accepted instead of s if we have

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

where θ_k is the threshold of the current iteration, with $\theta_k \geq 0$, $\theta_{k+1} \leq \theta_k$ and $\theta_\nu = 0$ if ν denotes the total number of elementary transformations performed.

This criterion avoids the computation of an exponential and the call to a random number generator, what usually saves 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 Koschnick [5] have related some convergence properties of threshold accepting methods to those of simulated annealing (but their proofs are not constructive).

It is quite easy to see that these thresholds θ_k can be seen as noises substracted from the variation $\Delta f(s,s')$ of f. Thus, threshold accepting algorithms can be considered as noising methods with the second way of perturbing f described above, and with a noise equal to $-\theta_k$ when iteration k is performed.

5 Applications of the noising methods

The first paper presenting a noising method [10] dealt with the following *clique partitioning problem*: given a complete non-oriented graph $G = (X, E)$ of which the edges are weighted by (negative or positive) integers, find a partition of X into $p(G)$ subsets so that the sum of the weights of the edges with their two extremities in the same subset is minimum. As explained above, in this noising method, the noises were added to the weights of the edges before applying a descent. This can be done for any problem represented by a weighted graph and, more generally, for problems with numerical data.

But sometimes the problem can be more "structural" than "numerical". It is one of the reasons that led us to design noising methods in which noises are added to the variations of f and not to the data. Another possibility to deal with such problems is that adopted by Bogdanova [7] for the following NP-hard problem, arising in coding theory. Let n and d be two positive integers; let Z be the set $\{0,1,2,3\}$ and let Z^n be the set of all n-tuples defined over Z. A set $C \subseteq Z^n$ is called a 4-ary M-code if $\mid C \mid = M$, and a 4-ary (n, M, d)-code if $\mid C \mid = M$ and if the Hamming distance δ between two distinct elements of C is at least d. The largest value of M such that a 4-ary (n, M, d)-code exists is denoted by $A_4(n, d)$. To determine lower bounds of $A_4(n, d)$, Bogdanova defines the function f^M over the

set of 4-ary M-codes as follows:

$$f^M(C) = \mid \{(x,y) \in C^2 \text{ such that } \delta(x,y) < d\} \mid$$

and, for a given value M, she tries to minimize f^M over the set of 4-ary M-codes C. If such a M-code C with $f^M(C) = 0$ is found, then $A_4(n,d) \geq M$. In this case, the same process is applied to f^{M+1}. In order to minimize f^M for any given value of M by the help of the noising method, Bogdanova gives a random value $k(v) \in [1-r, 1+r]$ to each element v of Z^n, where r is the arithmetically decreasing rate of noise. Then she defines the noised function f^M_{noised} over the set of 4-ary M-codes by:

$$f^M_{noised}(C) = \sum_{(x,y)\in C^2 \text{ and } \delta(x,y)<d} \frac{[k(x)+k(y)]}{2}.$$

Notice that, when the rate of noise r is equal to 0, $f^M_{noised} = f^M$. By means of this noising method, she succeeded to break several lower bounds of $A_4(n,d)$.

Anyway, even if the applications of the noising methods are not yet very numerous, it is not possible to detail all of them here. So we just summarize the field and references for the applications that we know:

- partitioning a weighted graph into cliques [10, 14, 15, 18, 19, 27, 28, 38];
- traveling salesman problem [2, 12, 16, 18, 30, 31];
- aggregation of linear orders into a median order (linear ordering problem) [13, 17-19];
- scheduling problems [4, 34, 35];
- covering and packing problems in coding theory: [7, 20, 21];
- 0-1 multidimensional knapsack problem [8, 26, 29];
- the multi-resource generalized assignment problem [3, 6];
- the prize-collecting Steiner tree problem [9];
- multi-criteria decision aid [11];
- the task allocation problem [22];
- the design of discrete manufacturing processes: [32];
- the alignment of graphemes and phonemes in linguistics: [40].

The noising methods involved in these applications may follow the basic noising method scheme developped in [10] as well as the two other types of noising f. Their results usually (but not always) show that they can compute good solutions, often better than those found by other metaheuristics as simulated annealing.

6 Conclusion

It would be unwise to conclude that the noising methods may solve any NP-hard problem efficiently. For instance, we tried to apply them to the computation of some Ramsey numbers, but we did not succeed to improve the currently best results.

Thus, the aim of this paper is only to show, through the principles and the applications of the noising methods, that this family of metaheuristics deserves interest because they are simple to implement and because they may provide good solutions within reasonable CPU times for quite different problems. There remain the problem of choosing the noising method scheme and, then, that of tuning the parameters of the chosen noising method. Some studies already mentioned [15, 18] show that it is possible to tune them automatically so that the user has just to give the desired CPU time before applying these methods.

Another subject which could be investigated in the future deals with the convergence of the noising methods. If we consider them as generalizations of simulated annealing or of threshold accepting methods, we benefit from the convergence results established for these metaheuristics (see for instance [1] for references about simulated annealing and [5] for threshold accepting methods): they show that there exist some noising method schemes (with an infinite number of iterations) which surely converge towards an optimal solution. Notice also that Johnson and Jacobson studied in [33] the convergence of the first noising type (noises added to the data): they give sufficient conditions to converge (still with an infinite number of iterations) towards an optimal solution.

These aspects of the noising methods (convergence and parameter tuning) as well as others such as their application to other combinatorial optimization problems or their hybridization with other metaheuristics (especially with genetic algorithms, as in [13]) will surely be the topics of our future research efforts devoted to this field.

7 References

[1] Aarts, E.H.L., and Lenstra, J.K. (eds.), Local search in combinatorial optimization, Wiley, New York, 1997.

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

[3] Alidaee, B., Amini, M., and Kochenberger, G.A., Hybrid Metaheuristic Approaches to the Multi-Resource Generalized Assignment Problem: a Case Study of Tabu Search, Space Smoothing, and Noising Methods, 2nd Metaheuristics International Conference (MIC-97), Sophia-Antipolis, 1997.

[4] Alidaee, B., Naidu, J.T., and Gillenwater, E.L., Heuristic Algorithms for Multiple Machine Scheduling with the Objective of Minimizing Total Weighted and Unweighted Tardiness, INFORMS 1997, Dallas, 1997.

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

[6] Amini, M.M., Alidaee, B., and Racer, M.J., Alternative Metaheuristics Approaches to the Multi-Resource Generalized Assignement Problem, INFORMS 1997, Dallas, October 1997.
[7] Bogdanova, G., Optimal codes over an alphabet of 4 elements, Proceedings of the Fifth international workshop on algebraic and combinatorial coding theory, Unicorn, Shumen, Sozopol, Bulgaria, 1996, 46-53.
[8] Boucher, P., and Plateau, G., Étude des méthodes de bruitage appliquées au problème du sac à dos à plusieurs contraintes en variables 0-1, Actes des 5es Journées nationales sur la résolution pratique de problèmes NP-complets (JNPC'99), Lyon, 1999, 151-162.
[9] Canuto, S. A., Ribeiro, and C. C., Resende, M. G. C., Local search with perturbations for the prize-collecting Steiner tree problem, Proceedings of the Third Metaheuristics International Conference, Angra dos Reis, 1999, 115-119.
[10] Charon, I., and Hudry, O., The noising method: a new combinatorial optimization method, Operations Research Letters 14, 1993, 133-137.
[11] Charon, I., and Hudry, O., An application of the noising method to MCDA, EURO XIV, Jerusalem, 1995.
[12] Charon, I., and Hudry, O., Mixing different components of metaheuristics, in: I.H. Osman and J.P. Kelly (eds.), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, Boston, 1996, 589-603.
[13] Charon, I., and Hudry, O., Lamarckian genetic algorithms applied to the aggregation of preferences, Annals of Operations Research 80, 1998, 281-297.
[14] Charon, I., and Hudry, O., Variations on the noising schemes for a clustering problem, Proceedings of the Third Metaheuristics International Conference, Angra dos Reis, 1999, 147-150.
[15] Charon, I., and Hudry, O., Noising methods for a clique partitioning problem, submitted for publication.
[16] Charon, I., and Hudry, O., Application of the noising method to the Travelling Salesman Problem, European Journal of Operational Research 125, 2000, 266-277.
[17] Charon, I., and Hudry, O., A branch and bound algorithm to solve the linear ordering problem for weighted tournaments, to appear in Discrete Applied Mathematics.
[18] Charon, I., and Hudry, O., Automatic tuning of the noising methods, submitted for publication.
[19] Charon, I., and Hudry, O., Descent with mutations for the aggregation of relations, submitted for publication.
[20] Charon, I., Hudry, O., and Lobstein, A., A new method for constructing codes, Proceedings of the 4th International Workshop on Algebraic and Combinatorial Coding Theory, Novgorod, 1994, 62-65.
[21] Charon, I., Hudry, O., and Lobstein, A., Application of the noising methods to coding theory, IFORS 96, Vancouver, 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] Dowsland, K. A., Simulated annealing, in C. Reeves (ed), Modern heuristic techniques for combinatorial problems, McGraw-Hill, London, 1995, 20-69.
[24] Dueck, G., and Scheurer, T., Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing, Journal of Computational Physics 90, 1990, 161-175.
[25] Dueck, G., and Wirsching, J., Threshold accepting algorithms for multi-constraint 0-1 knapsack problems, Technical paper TR 89 10 016, IBM Heidelberg Scientific Center, Germany, 1989.
[26] Fréville, A., Hanafi, S., and El Abdellaoui, A., Noising method for the 0-1 multidimensional knapsack problem, FRANCORO, Mons, 1995.
[27] Guillaume, D., and Murtagh, F., 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 and D. A. Roberts (eds.), Astronomical Data Analysis Software and Systems VIII, ASP Conference Series 172, ASP, San Francisco, 1999, 278-282.
[28] Guillaume, D., and Murtagh, F., Clustering of XML documents, in: Proceedings of From Information to Knowledge using Astronomical Databases, Computer Physics Communications, to appear.
[29] Hanafi, S., Fréville, A., and El Abdellaoui, A., Comparison of heuristics for the 0-1 multidimensional knapsack problem, in: I.H. Osman and J.P. Kelly (eds.), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, Boston, 1996, 449-465.
[30] Hwang, C.-P., Global and local search heuristics for the symmetric travelling salesman problems, PhD thesis, University of Mississippi, 1996.
[31] Hwang, C.-P., Alidaee, B., Johnson, J.D., A tour construction heuristic for the travelling salesman problem, Journal of the Operational Research Society 50, 1999, 797-809.
[32] Jacobson, S.H., Sullivan, K.A., and Johnson, A.W., Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms, Engineering Optimization 31, 1998, 247-260.
[33] Johnson A.W., and Jacobson, S.H., On the convergence of generalized hill climbing algorithms, submitted for publication.
[34] Kethley, R.B., 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, PhD thesis, University of Mississippi, 1997.
[35] Naidu, J., Alidaee, B., and Gillenwater, E., Heuristic Approaches to Minimize Tardiness Problems on Single Machine with Sequence Dependent Set-up Times, INFORMS 1997, Dallas, October 1997.
[36] Osman, I.H., and Kelly, J.P. (eds), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, Boston, 1996.
[37] Reeves, C. (ed), Modern heuristic techniques for combinatorial problems, McGraw-Hill, London, 1995.

[38] Sudhakar, V., and Siva Ram Murthy, C., A modified algorithm for the graph partitioning problem, Integration, the VLSI Journal 22, 1997, 101-113.
[39] Voss, S., Martello, S., Osman, I.H., and Roucairol, C., Meta-heuristics: Advances and trends in local search paradigms for optimization, Kluwer Academic Publishers, Boston, 1998.
[40] Yvon, F., Prononcer par analogie : motivation, formalisation et évaluation, PhD thesis, ÉNST, Paris, 1996.