ELSEVIER

European Journal of Operational Research 126 (2000) 1–12

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Invited Review

A framework for the description of evolutionary algorithms

Alain Hertz *, Daniel Kobler

Département de Mathématiques, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

Received 1 July 1999; accepted 16 September 1999

Abstract

Evolutionary algorithms (EA) are optimisation techniques inspired from natural evolution processes. They handle a population of individuals that evolve with the help of information exchange procedures. Each individual may also evolve independently. Periods of co-operation alternate with periods of self-adaptation. We define a terminology and give a general framework for the description of the main features of any particular evolutionary algorithm. Such a description does not provide, nor does it replace, algorithm pseudo-codes. The aim is to develop tools that may help understanding the "philosophy" of such methods. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Optimisation; Population-based methods; Evolutionary algorithms

1. Introduction

The techniques used to solve difficult combinatorial optimisation problems have progressively evolved from *constructive methods* to *local search techniques*, and finally to *population-based algorithms*. We draw a parallel between this evolution and the development of information exchange technologies. To illustrate this parallel, consider a researcher who is asked to solve a difficult optimisation problem.

- If he works alone and is not aware of any previous research linked to his problem, he will probably develop a constructive method. He will try to reproduce and to code in a systematic way what he would have done by hand.
- Assuming now that the researcher has read a paper devoted to the solution of his problem, he will take advantage of this previous work by generating a solution using the known technique. He will then concentrate his efforts in trying to improve this solution by means of a local search method.
- If our researcher has an easy access to libraries and to Internet, he can make a survey of the numerous existing methods that have been proposed for the solution of his problem. He can then generate a population of solutions by means of known constructive or local search methods. New solutions can easily be obtained by combining the best parts of the best solutions in the population, and improvement techniques

* Corresponding author. Tel.: +41-21-6932568; fax: +41-21-6934250.

E-mail address: alain.hertz@epfl.ch (A. Hertz).

PII: S0377-2217(99)00435-X

can be applied to each solution resulting from such combinations. By alternating combination periods with improvement periods, he will get a population-based method.

Population-based methods are very popular nowadays. They provide good solutions since any constructive method can be used to generate the initial population, and any local search technique can be used to improve each solution in the population. But population-based methods have the additional advantage of being able to combine good solutions in order to get possibly better ones. The basic idea behind this way of doing is that good solutions often share parts with optimal solutions.

Population-based methods are also called evolutionary algorithms (EAs). Many scientific papers have already been devoted to the design of new EAs or to the adaptation of a particular EA to a difficult combinatorial optimisation problem. Nevertheless, there is no clear definition of what an EA is. When researchers involved in this field are asked to define EAs, they give all kinds of answers. Some of them think that EAs are genetic algorithms. Others define EAs as solution techniques that handle an evolving population of solutions, and they therefore consider genetic algorithms as a particular EA.

The first answer does not take into account known population-based methods, such as ant systems, scatter search or adaptive memory algorithms. The second answer can be embarrassing. Indeed, assume that a local search technique is applied to two different solutions of a problem, and that these two processes are regularly interrupted in order to allow an exchange of information between both almost independent searches. While such an algorithm handles a population of two solutions (and can therefore be considered as an EA), it is probably closer to two independent runs of a local search technique.

This lack of precise definition creates some confusion in the OR community. For example, more and more OR papers are entitled "A genetic algorithm for ..." (Beasley and Chu, 1996; Chu and Beasley, 1996; Falkenauer and Delchambre, 1992; Fleurent and Ferland, 1996; Levine, 1994; Thangiah et al., 1991; Yamada and Nakano, 1992). Some of these papers use a local search technique as mutation operator, create offspring solutions by means of elaborated encodings and crossover operators, and use diversification strategies in order to avoid a premature convergence of the algorithm. We would rather define such an algorithm as an EA that includes an improvement algorithm for intensifying the search in some regions of the search space, as well as a diversification strategy that helps escaping a region of attraction in this search space.

Another drawback of this looseness of terminology is that very similar solution methods may have completely different names, depending on the preferences of the author. The risk of such a situation is that identical concepts are rediscovered several times under different names. This happened for example with basic ingredients of scatter search which have been rediscovered as new concepts that help improving a basic genetic algorithm.

The aim of this paper is to propose a terminology that should help describing the "philosophy", the main features of any particular EA. We first give in Section 2 the general scheme of an EA. This framework includes known solution techniques such as genetic algorithms, scatter search, ant systems and adaptive memory algorithms. In order to differentiate between all these EAs, we list in Section 2 some key elements that may help describing the philosophy of each particular EA. The proposed terminology is illustrated in Section 3 where we give the pseudo-code and the description of the main features of five known EAs. The last section contains concluding remarks.

2. Main features of population-based methods

Population-based methods are iterative solution techniques that handle a population of individuals and make them evolve according to some rules that have to be clearly specified. At each iteration, periods of *self-adaptation* alternate with periods of *co-operation*. Self-adaptation means that the individuals evolve independently while co-operation implies an exchange of information among the individuals. The following scheme

summarises a basic population-based method (see Fig. 1).

Many different algorithms can be described within this framework. For example, the selection and crossover operators of genetic algorithms can be seen as co-operation procedures while the mutation operator is part of the self-adaptation process.

For comparing two different EAs, we need to determine the main features that characterise each particular EA. These main features are described below. Most of them are also briefly described in Calégari et al. (1999). In order to illustrate the various forms that these features can take, we will consider several EAs that have been proposed for the solution of the *k-Colouring Problem* and the *Vehicle Routing Problem*. In the *k*-Colouring Problem, the objective is to colour the vertices of a graph with a given number *k* of colours in such a way that no two adjacent vertices have the same colour. The Vehicle Routing Problem is to specify the routes of a fleet of vehicles delivering goods to clients. The objective is to minimise the total length of the routes, while taking vehicle capacities into account.

2.1. Description of the individuals

While local search techniques iteratively try to improve a single solution, EAs handle a population of individuals. These individuals are not necessarily solutions of the considered problem. They may be parts of a solution. They can even represent any kind of object with the property that there exists a procedure which is able to transform these objects into solutions of the considered problem.

To illustrate this concept, consider first the Vehicle Routing Problem. Most EAs that have been proposed for the solution of this problem define individuals as (possibly infeasible) solutions of the problem. However, Taillard (Golden et al., 1997; Rochat and Taillard, 1995) has recently proposed an EA in which individuals are defined as vehicle routes. Specific procedures have been designed for creating a feasible solution by combining different vehicle routes. As second example, we mention the ant system described in Zufferey and Hertz (1997) for the *k*-Colouring Problem. Individuals are defined as coloured ants that move along the edges of the graph, and a procedure has been developed for generating a colouring of a graph on the basis of the location of the ants on this graph. A similar model has been proposed in Kuntz and Snyers (1994) for a graph partitioning problem.

```
Generate an initial population of individuals;
While no stopping condition is met do
    co-operation;
    self-adaptation;
EndWhile.
```

Fig. 1. General scheme of a population-based method.

While individuals can be pieces of solutions that have to be gathered in order to create a complete solution, individuals can also represent sets of solutions. This is the case, for example, in island-based genetic algorithms (Cantú-Paz, 1995) where solutions are grouped together in order to form *islands* of solutions. During the co-operation periods of this EA, each island makes available any information contained in it. Solutions can move from one island to the other. In the self-adaptation periods, each island is modified independently by means of a basic genetic algorithm in which individuals are solutions of the considered problem.

The definition of the link between individuals and solutions of the considered problem gives indication about the levels at which information exchange occurs.

2.2. Evolution process

The population of an EA iteratively evolves according to some specific rules. At each iteration, new individuals are created and it has to be decided which ones will enter the population. Moreover, individuals must be thrown out of the population in order to avoid a continuous expansion of the population. From one iteration to the next, the population can be totally changed in that sense that only new individuals are kept in the population. Such a strategy is called *generational*

replacement. On the other hand, a *steady-state evolution* means that only part of the population can be modified between two consecutive iterations.

Most EAs work with populations of fixed size p, and it is usually decided that the p best individuals survive for the next iteration. The population size may however also vary during the search process. Consider for example a population containing old individuals as well as newly generated ones. Deciding which one will be kept in the population for the next iteration can be done on a random basis. Such a probabilistic decision process produces unpredictable outcomes, and one can prefer to decide that an individual stays in the population if its value (measured by some function) is close enough to the best value in the population (for example, within 5%).

It is important to mention that EAs are parallel in essence. This certainly explains why implementations of EAs on parallel computers are thoroughly investigated. Parallel programming allows *asynchronous evolution*. In such a case, each individual is continuously modified (independently or by means of information exchanges) without checking whether changes have already been performed on the other individuals.

2.3. Neighbourhood structure

Two individuals in the population are not necessarily allowed to exchange information during the co-operation phase. Indeed, it can be specified that each individual is associated with a subset of individuals which may transmit information about their value, their contents, etc. In such a case, the population is said to be *structured*. This structure can be summarised by a neighbourhood function N that assigns a proper subset of the population to each of its members.

By defining this function N in such a way that, for every individual i, $N(i) \cup \{i\}$ is equal to the whole population, one gets an *unstructured* population where each individual may communicate with each other.

Structured populations have been used in many adaptations of EAs to particular optimisation problems. The most common structures are rings (Laszewski, 1991; Gordon and Whitley, 1993; Mühlenbein, 1989), grids (Manderick and Spiessens, 1989), and even more complex ones such as hypercubes (Tanese, 1989).

2.4. Information sources

When an individual i has a large set $N(i)$ of neighbours with which he is allowed to communicate, this does not mean that he will take advantage of all this large amount of available information. He will perhaps choose only one neighbour at random that will co-operate with him in order to create a new individual. This occurs in genetic algorithms where *offspring* individuals are created on the basis of only two *parent* individuals. However, the number of individuals that co-operate in order to create new ones may be larger than two. This is the case in a scatter search. Indeed, this optimisation technique does not impose a limit of two on the number of individuals that may be linearly combined.

In the same spirit, the adaptive memory algorithm proposed in Rochat and Taillard (1995) for the Vehicle Routing Problem does not impose any restriction on the number of vehicle routes that may contribute to the creation of a new solution. This EA considers each vehicle route as an individual. The combination of some vehicle routes leads to the creation of a solution which is improved by means of a local search technique. This improved solution is then redecomposed into vehicle routes that are considered as candidates for entering the population.

Notice that when a set of individuals are co-operating in order to create a new one, this does not mean that each of these individuals transmits information. For example, unilateral exchanges occur in an island-based genetic algorithm (Cantù-Paz, 1995; Gordon and Whitley, 1993). Indeed, this EA defines individuals as sets of solutions called islands. These islands are located on a directed ring, and they regularly transmit their best solution to their successor. Hence, when two individuals co-operate, one of them gives the information while the other is appropriately modified.

In such a case, only one parent has been used to modify an individual.

On the contrary, the whole population may contribute to the creation of a new individual. Even more, all populations encountered during the previous iterations may take part to this information exchange. In such a case, the creation of new individuals is said to be based on the *history* of the population. By history, we mean information that cannot be obtained through the analysis of the individuals in the current population. A historical account of the populations of the previous iterations is needed for making such an information available. Ant systems are strongly based on this concept since the pheromone trail used to guide the ants is a kind of summary of what happened since the beginning of the search process.

2.5. Infeasibility

Each EA is based on a precise definition of the individuals. When combining information originating from a set of individuals, it may occur that the new created object is not a feasible individual. Consider for example two feasible colourings C_1 and C_2 of a graph G. A new colouring may be obtained by randomly choosing in C_1 or C_2 the colour to be assigned to each vertex of G. Such a combination may lead to infeasible colourings in which adjacent vertices have the same colour.

This concept can also be illustrated with the Vehicle Routing Problem. If two feasible solutions exchange some parts of their vehicle routes, it may occur that clients are visited more than once, some may be never visited, and vehicle capacity constraints may be violated. It is then important to specify how infeasibility is dealt with.

As mentioned in Liepens and Potter (1991), at least three strategies can be followed. A simple way which has not proven to be successful is to simply *reject* infeasible individuals obtained during the co-operation phase. A second approach is to accept infeasibility and *penalise* it in the function that measures the quality of an individual. The third alternative is to develop specialised combination procedures that can only create feasible individuals. This last approach often uses a *repair* procedure that transforms infeasible individuals into feasible ones.

For example, consider a scatter search in which individuals are integer vectors (Glover, 1977). Linear combinations of integer vectors may produce infeasible individuals. In such a case, integer values are obtained by means of a rounding process that is applied on each component of the new generated vectors.

2.6. Intensification strategy

It has been observed that many EAs can significantly be improved by using advanced *improvement algorithms* during the self-adaptation phase. By improvement algorithm, we mean any procedure that tries to improve the value of an individual without using information originating from other individuals. The use of an improvement algorithm aims to intensify the search in some regions of the search space. Embedding a local search technique within an EA has proven to be successful in many applications (Costa et al., 1995; Fleurent and Ferland, 1996; Levine, 1994). This is not surprising. Indeed, while the use of a population of individuals ensures an exploration of a large part of the search space, improvement algorithms help to determine high quality individuals in identified good regions of the search space.

The success of an EA that does not use any improvement algorithm can sometimes be explained by the transmission of pertinent information during the co-operation phase. Indeed, for certain problems, it is possible to relate the good quality of an individual with a particular part of the information contained in it. In such a case, new individuals of good quality can easily be generated by mixing adequate pieces of information from the best known individuals. For example, it seems reasonable to assume that good solutions of a Vehicle Routing Problem have vehicle routes in common with optimal solutions. Hence, the combination of the vehicle routes of the best known solutions may lead to the creation of even better solutions. This can explain the success of EAs in

which individuals are defined as vehicle routes rather than feasible solutions of the Vehicle Routing Problem.

The situation is more complicated for the k-Colouring Problem. Indeed, it is difficult to precisely locate the reason why a colouring of a graph is better than another one. In such a case, improvement algorithms are necessary in order to ensure the success of the considered EA.

2.7. Diversification strategy

One of the major difficulties observed when using EAs is the premature convergence of the algorithm towards local optima. In order to steer individuals away from local optima, a *noise* procedure may be used, that randomly perturbs the individuals. Such a procedure modifies each individual independently, but contrarily to improvement algorithms, it has unexpected results on the value of an individual, in the sense that it does not necessarily improve it. The most famous example of noise procedure is the mutation operator in genetic algorithms.

A different philosophy can be used to move away from individuals that may cluster around complex regions of attraction. Instead of using noise that produces unpredictable outcomes, a diversification strategy can be applied in order to systematically generate new individuals that lie in new regions. Scatter search uses such a diversification strategy (Glover, 1994, 1977). The cooperation mechanism is based on non-convex combinations of vectors (i.e., individuals). Such a strategy can therefore extrapolate beyond the region spanned by the combined individuals.

2.8. Summary of the main features

To summarise the above concepts, consider any EA designed for the solution of a particular optimisation problem. The "philosophy" of this EA can be described by means of the following main features.

Individuals. The relation between the individuals and the considered problem must be specified. Individuals can be feasible or infeasible solutions of a problem. They can be parts of solutions, sets of solutions, or anything else.

Evolution process. The population can be of fixed size or not. It must be specified whether the population evolves according to a generational replacement or a steady state. In case of an implementation of the EA on a parallel computer, asynchronous evolution can occur.

Neighbourhood. Information exchange can be based on a structured or unstructured population. In the first case, the special structure that is used can be specified (e.g., ring, grid, hypercube).

Information sources. The number of parents needed to create new individuals must be specified. This number can be equal to any strictly positive integer. New individuals can also be created on the basis of the history of the population.

Infeasibility. The way infeasibility is dealt with must be specified. Infeasible individuals can be rejected, penalised or repaired.

Intensification. During the self-adaptation phase, an improvement algorithm can be applied to each individual in the population.

Diversification. In order to avoid premature convergence, a noise procedure can be applied to each individual. A more elaborate diversification strategy can also be used.

We have listed the main features of EAs, as well as possible forms that these features can take. The list is of course not exhaustive. It should evolve in concert with the possible evolution of EAs.

3. Illustration

We now illustrate the use of the above concepts for the description of various EAs. It is important to note that the main feature description is not a substitute for the pseudo-code. They should really both be given in order to have a complete summary of the method, which neither does on its own.

3.1. Genetic algorithms

As first example, we consider a basic *genetic algorithm* as described in Davis, (1991) and

Goldberg (1989). Genetic algorithms are inspired by Biology and in particular by those biological processes that allow populations of organisms to adapt to their surrounding environment. Pioneering work on the development of this method date back to the mid-1960s (Fogel et al., 1966; Holland, 1962; Rechenberg, 1965).

A genetic algorithm works with three operators that are iteratively used in order to make the population evolve. The *selection* operator decides which individuals may survive in the population. Good individuals have usually more chance to be selected. In can be imposed that the best one will survive. The *crossover* operator combines two individuals in order to create two new ones called *offspring* individuals. Selection and crossover occur at the co-operation phase while the third operator, called *mutation*, is applied at the self-adaptation phase. The mutation operator introduces some noise in the population in order to prevent premature convergence towards local optima.

A genetic algorithm with generational replacement can be described either by the following pseudo-code or by specifying the particular form of each main feature, as shown in Fig. 2.

3.2. Scatter search

As second example, we consider the *scatter search* proposed by Glover (1994, 1977). This EA is a search strategy that systematically generates a set of *dispersed points* from a chosen set of *reference points*. This is done by making linear combinations of subsets of the current reference points. Points resulting from such combinations are called *trial points*. Instead of using only non-negative linear combinations of points, as in the approaches of Lagrangian and surrogate relaxations, scatter search uses more general linear combinations that include negative weights. Such combinations can therefore produce points both inside and outside the convex region spanned by the reference points.

While points correspond to feasible solutions of the considered problem, trial points may violate some constraints. These trial points are therefore modified by means of a repair procedure that transforms them into feasible points. An improvement algorithm is then applied on each new point in order to try to generate points of still higher quality. These improved points form the set of dispersed points. The new set of reference points that will be used at the next iteration is selected among the current reference and dispersed points. This selection as well as the creation of trial points occur at the co-operation phase while the repair procedure and the improvement algorithm are used at the self-adaptation phase.

Fig. 3 contains the pseudo-code as well as the description of the main features of a basic scatter search.

Choose an even integer $p \geq 2$ and generate an initial population P_0 of p individuals;
Set i := 0; (iteration counter)
While no stopping criteria is met **do**
    set i := i +1 and initialise P_i to the empty set;
    While P_i has less than p individuals **do**
        Select two individuals I_1 and I_2 in P_{i-1};
        Apply the crossover operator to I_1 and I_2 for creating offspring O_1 and O_2;
        Add O_1 and O_2 to P_i;
    EndWhile
    Apply the mutation operator to each individual in P_i;
EndWhile

Individuals	:	usually, feasible solutions.
Evolution process	:	generational replacement; population of constant size.
Neighbourhood	:	possibly structured.
Information sources	:	two parents.
Infeasibility	:	never occurs.
Intensification	:	none.
Diversification	:	noise procedure (mutation).

Fig. 2. Description of a basic genetic algorithm with generational replacement.

Generate an initial set R_0 of reference points;
Set i := 0; (iteration counter)
While no stopping criteria is met **do**
    Set i := i +1;
    Determine a set T_i of trial points by making linear combinations of points in R_{i-1};
    Transform the trial points in T_i into a set F_i of feasible points;
    Improve the points in F_i in order to obtain a set D_i of dispersed points;
    Select points in $R_{i-1} \cup D_i$ to be put in the new set R_i of reference points;
EndWhile

Individuals	:	feasible solutions of the problem.
Evolution process	:	steady state; population of usually constant size.
Neighbourhood	:	unstructured.
Information sources	:	at least two parents.
Infeasibility	:	repaired.
Intensification	:	improvement algorithm.
Diversification	:	non-convex combination of individuals.

Fig. 3. Description of a basic scatter search.

3.3. Ant systems

The third illustration is given by *ant systems*. This search method is inspired by biological observations on the behaviour of ant colonies. In this EA, simple agents, called *ants*, explore a region of the search space and share information gathered throughout their individual searches. Introduced in 91 by Colorni et al. (1991, 1992), ant systems are now widely used for the solution of various combinatorial optimisation problems (Colorni et al., 1994; Costa and Hertz, 1997; Dorigo and Gambardella, 1997; Gambardella and Dorigo, 1996; Maniezzo et al., 1994).

In a basic ant system, each ant is a constructive procedure that is able to generate a solution to the considered problem. At each step of the construction, each ant has to decide how to make one more step towards the completion of a partial solution. Such a choice is based on two factors. The *trace factor* guides the ants to choices that gave good results in earlier constructions. The intensity of this factor informs the ants about the quality of the solutions that have been generated by making the corresponding choice. The *desirability factor* guides the ants to choices which induce the best value of a function that measures the quality of a partial solution.

Once all ants have completed their construction, the trace factors are updated. A choice that led to good solutions induces an increase of the corresponding trace factor, while bad choices induce a decrease of this factor. The solutions constructed by the ants are further submitted to an improvement algorithm. The use of the trace factor occurs at the co-operation phase while the desirability factor and the improvement algorithm are used at the self-adaptation phase.

We give in Fig. 4 both the pseudo-code and the description of the main features of a basic ant system.

Initialise the trace factors and choose a number n of ants;
While no stopping criteria is met **do**
  Build n new solutions using the desirability and trace factors;
  Update the trace factors;
  Apply an improvement algorithm on each new solution;
EndWhile

Individuals	:	feasible solutions obtained by means of a constructive algorithm.
Evolution process	:	generational replacement; population of constant size.
Neighbourhood	:	unstructured.
Information sources	:	history of the evolution.
Infeasibility	:	never occurs.
Intensification	:	improvement algorithm.
Diversification	:	none.

Fig. 4. Description of a basic ant system.

3.4. Adaptive memory algorithms

Taillard has recently developed an extension of tabu search that allows automatic diversification and intensification of the search process (Golden et al., 1997; Rochat and Taillard, 1995; Taillard et al., 1997). This new method, called *adaptive memory algorithm*, can be considered as an EA that works as follows. A central memory is in charge of keeping track of the best components of the solutions visited during the search. These components are combined in order to create new solutions. If these new solutions are not feasible, they are submitted to a repair procedure. They are then improved by means of a tabu search, or any improvement algorithm. Finally, the components of the new solutions are considered as candidates that may be selected for replacing old components in the central memory.

The improvement algorithm is used at the self-adaptation phase while the combination procedure and the selection process occur at the co-operation phase.

During the first iterations, the central memory contains very different components and the combination procedure will therefore have the tendency to create a diversity of new solutions. Hence, diversification occurs during the initial steps. Then, the components contained in the central memory will tend to be parts of a very small set of solutions located in a limited number of regions of the search space. The search process moves therefore gradually from diversification to intensification. This is clearly shown in Rochat and Taillard (1995).

The pseudo-code and the description of the main features of a basic adaptive memory algorithm are given in Fig. 5.

Generate a set of solutions and introduce their components in the central memory;
While no stopping criteria is met **do**
    Combine components of the central memory in order to create new solutions;
    Use a repair procedure on each infeasible new solution;
    Apply an improvement algorithm on each new solution;
    Update the central memory by removing some components and introducing new ones originating from the new feasible solutions;
EndWhile

Individuals	:	components of feasible solutions.
Evolution process	:	steady state; population of fixed size.
Neighbourhood	:	unstructured.
Information sources	:	at least two parents.
Infeasibility	:	repaired.
Intensification	:	implicit intensification during the last iterations; improvement algorithm.
Diversification	:	implicit diversification during the first iterations.

Fig. 5. Description of a basic adaptive memory algorithm.

3.5. Island-based genetic algorithms

As already mentioned, EAs can deal with individuals that do not necessarily correspond to feasible solutions. Individuals can be parts of solutions, sets of solutions, or any kind of information that can be combined in order to create feasible solutions.

Island-based genetic algorithms (Cantù-Paz, 1995; Gordon and Whitley, 1993) illustrate the case where individuals are sets of solutions. This EA works at two different levels. At the *ground level*, individuals are defined as feasible solutions of a problem. These individuals evolve by means of a genetic algorithm. At the *top level*, individuals are defined as sets of solutions, called *islands*, and are located on a directed ring. Each island regularly transmits its best solution to its successor on the ring.

We give in Fig. 6 the pseudo-code and the description of the main features of a steady-state island-based genetic algorithm that uses an improvement algorithm at the ground level.

3.6. A case study

We finally give the pseudo-code and the description of the main features of an EA that has been proposed by Fleurent and Ferland (1996) for the solution of the k-Colouring Problem. We show how such a description can help understanding the philosophy of the proposed algorithm.

Fleurent and Ferland propose to solve the k-Colouring Problem by means of a steady state genetic algorithm that uses a tabu search as improvement algorithm. Their EA is one of the best known heuristic for the k-Colouring Problem. It is described in Fig. 7.

Choose the number k of islands and the size p of each island;
Generate a set of k*p initial feasible solutions and partition this set into k islands P_1,..., P_k;
Set i := 0; (iteration counter)
While no stopping criteria is met **do**
    Set i := i+1;
    For each island P_j **do**
        Select two solutions I_1 and I_2 in P_j;
        Apply the crossover operator to I_1 and I_2 for creating offspring O_1 and O_2;
        Apply the mutation operator and the improvement algorithm on O_1 and O_2;
        Decide whether or not O_1 and O_2 should enter P_j for replacing older solutions;
    EndFor
    If i is multiple of a given integer n **then**
        Move the best solution of each island P_j to island $P_{(j \bmod k)+1}$
EndWhile

Ground level

Individuals	:	feasible solutions.
Evolution process	:	steady state; population of constant size.
Neighbourhood	:	unstructured.
Information sources	:	two parents.
Infeasibility	:	never occurs.
Intensification	:	improvement algorithm.
Diversification	:	noise procedure.

Top level

Individuals	:	sets of feasible solutions.
Evolution process	:	generational replacement; population of constant size.
Neighbourhood	:	structured on a ring.
Information sources	:	one parent.
Infeasibility	:	never occurs.
Intensification	:	none.
Diversification	:	none.

Fig. 6. Description of a steady-state island-based genetic algorithm.

Generate an initial population P of feasible solutions;
While no stopping criteria is met **do**
    Select four individuals I_1, I_2, I_3 and I_4 in P;
    Apply the crossover operator to I_1 and I_2 for creating offspring O_1;
    Apply the crossover operator to I_3 and I_4 for creating offspring O_2;
    Apply a tabu search on O_1 and O_2;
    Insert O_1 and O_2 in P;
    Remove the two worst solutions in P;
EndWhile

Individuals	:	partitions of the vertex set into k subsets.
Evolution process	:	steady state; population of constant size. at most two individuals are modified at each iteration. the combination procedure uses the graph structure.
Neighbourhood	:	unstructured.
Information sources	:	two parents.
Infeasibility	:	never occurs.
Intensification	:	tabu search.
Diversification	:	the selection of parent individuals is based on the entropy of the population.

Fig. 7. Description of the EA proposed in Fleurent and Ferland (1996) for the k-Colouring Problem.

Some main features of this EA are described in more details when compared to the previous figures. For example, we have stressed that only two individuals can evolve at each iteration while a tabu search is used as improvement algorithm. Such a precise description helps observing that most of the success of this EA is due to the improvement algorithm. The authors confirm this fact. Their method is in fact an extension of a tabu search that assures a diversified exploration of the search space. This does not clearly appear in the pseudo-code.

We have also pointed out that the combination procedure uses the graph structure and is therefore far from the 1-point, 2-point and uniform crossovers typically used in genetic algorithms. While the paper is entitled "Genetic and hybrid algorithms for graph colouring", we think that the method has almost nothing in common with a basic genetic algorithm. Indeed, the authors use an unusual crossover operator, while the mutation operation is replaced by a tabu search. This is exactly what we meant in the introduction when we observed that many papers entitled "A genetic algorithm for ..." describe extensions of genetic algorithms that should be called evolutionary algorithms or population-based methods.

Notice that we did not give any implementation details in the description of the main features. We had in mind to point out the key elements that may help understanding the philosophy of the proposed EA.

4. Final remarks

The main features described in Section 2 and illustrated in Section 3 can be used as basic ingredients for a classification scheme. For example, Calégari et al. (1997) propose to represent any particular EA in the following compact form:

(0)[(1)(2)](3)(4)(5)[(6)(7)(8)],

where each entry (*i*) $0 \leqslant i \leqslant 8$ has the following contents.

(0) short description of the individuals
(1) *Y* if the population has a constant size, and *N* otherwise
(2) *Y* if the population is structured, and *N* otherwise
(3) number of parents if it is fixed. The abbreviation h_s is added to this number if the evolution process uses the history of the population
(4) *nvr* if infeasibility never occurs, *pen* if it is penalised, *rep* if it is repaired, and *die* if infeasible individuals are thrown out of the population
(5) *Y* if each individual has some evolving information attached to it that indicates how it behaves in certain situations, and *N* otherwise
(6) *Y* if an improvement algorithm is used, and *N* otherwise
(7) *Y* if a noise procedure is used, and *N* otherwise
(8) *gr* in case of a generational replacement, *ss* in case of steady state, and *as* if asynchronous evolution occurs

For example, the basic scatter search described in Fig. 3 can be summarised in the following compact form:

(feasible solutions)[YN]_repN[YNss].

Such a compact description can be compared to the classification schemes proposed in Blazewicz et al. (1983), Blazewicz et al. (1993) and Graham et al. (1979) for scheduling problems.

The current interest in EAs is more brought to their implementation rather than to the algorithmic mechanism. We believe that the decision of using or not a given ingredient is more important than the design of the way this ingredient is practically implemented. This is the reason why we did not mention, for example, how individuals are encoded. Once it has been decided which kind of information will be exchanged during the cooperation phase of an EA, then any encoding can be chosen as long as it facilitates such exchanges.

Consider for example the *k*-Colouring Problem. If the colour of each vertex is the information that should be transmitted, then individuals can be defined as strings whose components are the colours of the vertices. Such a string-based encoding has been used in Fleurent and Ferland (1996) for

example. However, one might prefer to combine colour classes. In such a case, each individual can be encoded as a partition of the vertex set into colour classes. Such an encoding has recently been proposed by Falkenauer and Delchambre (1992) and Falkenauer (1996).

As shown in Section 3.6, the description of the main features of an EA can help understanding the philosophy of the method. It can also serve to better analyse the reasons that explain the good performance of a particular EA. For example, an EA that does not use any improvement algorithm certainly owes its success to the information exchange process. On the other hand, if a powerful local search technique is used as intensification strategy, the advantage of using a population-based method should be clearly proved.

References

Beasley, J.E., Chu, P.C., 1996. A genetic algorithm for the set covering problem. European Journal of Operational Research 94, 393–404.

Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.G., 1983. Scheduling subject to resource constraints: Classification. Discrete Applied Mathematics 5, 11–24.

Blazewicz, J., Ecker, K., Schmidt, G., Weglarz, J., 1993. Scheduling in Computer and Manufacturing Systems. Springer, Berlin.

Calégari, P., Coray, G., Hertz, A., Kobler, D., Kuonen, P., 1997. Evolutionary algorithm revisited: the TEA Classification. Technical Report ORWP 97/02, Swiss Federal Institute of Technology, Lausanne, Switzerland.

Calégari, P., Coray, G., Hertz, A., Kobler, D., Kuonen, P., 1999. A taxonomy of evolutionary algorithms in combinatorial optimization. Journal of Heuristics 5, 145–158.

Cantù-Paz, E., 1995. A summary of research on parallel genetic algorithms. Technical Report, Illinois Genetic Algorithms Laboratory, USA.

Chu, P.C., Beasley, J.E., 1996. A genetic algorithm for the multidimensional knapsack problem. Technical Report, Imperial College, London, UK.

Colorni, A., Dorigo, M., Maniezzo, V., 1991. Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, Bradford Books. MIT Press, Cambridge, MA, pp. 134–142.

Colorni, A., Dorigo, M., Maniezzo, V., 1992. An investigation of some properties of an ant algorithm. In: Männer, R., Manderick, B. (Eds.), Proceedings of the Second European Conference on Parallel Problem Solving from Nature. Elsevier, Amsterdam, pp. 509–520.

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., 1994. Ant system for job shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 34, 39–53.

Costa, D., Hertz, A., Dubuis, O., 1995. Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs. Journal of Heuristics 1, 105–128.

Costa, D., Hertz, A., 1997. Ants can colour graphs. Journal of the Operational Research Society 48, 295–305.

Davis, L., 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1, 53–66.

Falkenauer, E., Delchambre, A., 1992. A genetic algorithm for bin packing and line balancing. In: Proceedings of the 1992 IEEE International Conference on Robotics and Automation. IEEE Computer Society Press, Silver Spring, MD, pp. 1186–1192.

Falkenauer, E., 1996. A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics 2, 5–30.

Fleurent, C., Ferland, J.A., 1996. Genetic and hybrid algorithms for graph coloring. In: Laporte, G., Osman, I.H. (Eds.), Metaheuristics in Combinatorial Optimization. Baltzer Science Publishers; Annals of Operations Research 63, 437–461.

Fogel, L.J., Owens, A.J., Walsh, M.J., 1966. Artificial Intelligence Through Simulated Evolution. Wiley, New York.

Gambardella, L.M., Dorigo, M., 1996. Solving symmetric and asymmetric TSPs by ant colonies. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation. IEEE Press, New York, pp. 622–627.

Glover, F., 1977. Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166.

Glover, F., 1994. Genetic algorithms and scatter search: Unsuspected potentials. Statistics and Computing 4, 131–140.

Goldberg, D., 1989. Genetics Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Golden, B.L., Laporte, G., Taillard, E.D., 1997. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. Computers and Operations Research 24, 445–452.

Gordon, V., Whitley, D., 1993. Serial and parallel genetic algorithms as function optimizers. In: Forrest, S. (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann, Los Altos, CA, pp. 177–183.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling theory: A survey. Annals of Discrete Mathematics 5, 287–326.

Holland, J.H., 1962. Outline for a logic theory of adaptative systems. Journal of the ACM 3, 297.

Kuntz, P., Snyers, D., 1994. Emerging coloration and graph partitioning. In: Proceedings of the Third International

Conference of Adaptative Behavior. MIT Press, Cambridge, MA, pp. 494–500.

Laszewski, G. von., 1991. Intelligent structural operators for the k-way graph partitioning problem. In: Belew, R., Brooker, L. (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, Los Altos, CA, pp. 45–52.

Levine, D., 1994. A parallel genetic algorithm for the set partitioning problem, Ph.D. thesis. Illinois Institute of Technology, USA.

Liepens, G.E., Potter, W.D., 1991. A genetic algorithm approach to multi-fault diagnosis. In: Davis, L. (Ed.), Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, pp. 237–250.

Manderick, B., Spiessens, P., 1989. Fine-grained parallel genetic algorithms. In: Schaffer, J.D. (Ed.), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, Los Altos, CA, pp. 428–433.

Maniezzo, V., Colorni, A., Dorigo, M., 1994. The ant system applied to the quadratic assignment problem. Technical Report 94/28, IRIDIA, Université Libre de Bruxelles, Belgium.

Mühlenbein, H., 1989. Parallel genetic algorithms, population genetics and combinatorial optimization. In: Schaffer, J.D. (Ed.), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, Los Altos, CA, pp. 416–421.

Rechenberg, I., 1965. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment Transl. 1122, B.F. Toms Transl. Ministry of Aviation, Royal Aircraft Establishment, Farnborough, Hants, UK.

Rochat, Y., Taillard, E.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167.

Taillard, E.D., Gambardella, L.-M., Gendreau, M., Potvin, J.-Y., 1997. Programmation à mémoire adaptative, technical report IDSIA-79–97, IDSIA, Lugano, Switzerland.

Tanese, R., 1989. Distributed genetic algorithms. In: Schaffer, J.D., Philips Laboratories, (Eds.), Proceedings of the Second International Conference on Genetic Algorithms. Morgan Kaufmann, Los Altos, CA.

Thangiah, S.R., Nygard, K.E., Juell, P.L., 1991. GIDEON: A genetic algorithm system for vehicle routing with time windows. In: Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications. IEEE Computer Society Press, Silver Spring, MD, pp. 322–328.

Yamada, T., Nakano, R., 1992. A genetic algorithm applicable to large-scale job-shop problems. In: Männer, R., Manderick, B. (Eds.), Proceedings of the Second European Conference on Parallel Problem Solving from Nature. Elsevier, Amsterdam, pp. 281–290.

Zufferey, N., Hertz, A., 1997. Coloration de graphes à l'aide de fourmis, Technical Report. Swiss Federal Institute of Technology, Lausanne, Switzerland.