Spacing Memetic Algorithms

Daniel Cosmin Porumbel
Univ. Lille Nord de France,
F-59000 Lille, France
UArtois, LGI2A, F-62400,
Béthune, France
daniel.porumbel@univ-artois.fr

Jin-Kao Hao
Univ. Angers, LERIA, 2 Bd Lavoisier, 49045 Angers
hao@info.univ-angers.fr

Pascale Kuntz
Univ. Nantes, LINA, BP 50609, 44306 Nantes, France
pascale.kuntz@univ-nantes.fr

ABSTRACT

We introduce the Spacing Memetic Algorithm (SMA), a formal evolutionary model devoted to a systematic control of spacing (distances) among individuals. SMA uses search space distance information to decide what individuals are acceptable in the population, what individuals need to be replaced and when to apply mutations. By ensuring a "healthy" spacing (and thus diversity), SMA substantially reduces the risk of premature convergence and helps the search process to continuously discover new high-quality search areas. Generally speaking, the number of distance calculations represents a limited computational overhead compared to the number of local search iterations. Most existing memetic algorithms can be "upgraded" to a spacing memetic algorithm, provided that a suitable distance measure can be specified. The impact of the main SMA components is assessed within several case studies on different problems.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—Heuristic methods

General Terms

Algorithms, Optimization, Search Space

Keywords

distance measure, spacing, diversity, memetic algorithms

1. INTRODUCTION AND MOTIVATIONS

Memetic algorithms [16] represent a well-established approach to large and hard optimization problems. Typically, a memetic algorithm (MA) is an evolutionary algorithm (EA) that integrates local search. The objective of the local operator is to search for higher-quality individuals in the proximity of the offspring solutions generated by recombination.

Diversity control has always attracted interest in evolutionary computing. Consequently, the design of an effective EA typically employs, explicitly or implicitly, certain diversification strategies. However, the concept of population diversity has also suffered changes over time, and has been un-synonymously associated to different measures and indicators, e.g., statistical dispersion, genetic entropy, fitness variance, or multi-objective diversity—see more examples in [1]. In our memetic framework, we capture the notion of diversity with *specific spacing indicators* based on distances between individuals.

SMA proposes modifications on a *few* points of the classical MA template. First, an advanced offspring rejection procedure is introduced: based on distance information, the offspring can be accepted, rejected or mutated (Section 2.2). Secondly, SMA integrates a survival selection approach guided by both *spacing and fitness* (Section 2.3). We also propose advanced extensions (e.g., reactive dispersion) that strive to unlock the search from more challenging landscape plateaux or "traps" (Section 2.4).

By coupling the above components, SMA aims at fulfilling two clearly-defined spacing objectives (Section 2.1.1) *without* "sacrificing quality for diversity". Furthermore, these spacing components can be "attached" to an existing MA without any change on the internal MA operators (SMA re-uses the MA crossover, local search, parent and survival selection, etc.). Besides introducing new techniques in memetic computing, SMA unifies several interesting ideas developed over time in rather disparate contexts (Section 3)—e.g., offspring addition tests in discrete optimization [20, 21], niching and crowding in multimodal optimization [15, 4, 2, 19], distance-guided recombination [8, 6], and others [23, 9, 5].

The next section is devoted to a formal description of the Spacing Memetic Algorithm. Section 3 explores related ideas from the literature and discusses their differences compared to our approach. In section 4, we assess the impact of the main SMA components on several case studies (on artificial problems, maximum clique, and graph coloring), followed by conclusions in the last section.

2. THE PROPOSED SPACING MEMETIC ALGORITHM (SMA)

2.1 Main objectives and algorithmic template

2.1.1 Distance measure and spacing definitions

We consider a population of individuals $Pop = \{I_1, I_2, \ldots, I_{|Pop|}\}$ from a search space Ω; we assume our objective is to minimize the fitness function f over Ω. We also consider a *search space distance* d, i.e., a reflexive and symmetric function $d : \Omega \times \Omega \to I\!R_+$. A meaningful distance measure should respect certain correlation and proximity properties with regard to the neighborhood and to the landscape of the problem.

In our context, $d(X_1, X_2)$ is equivalent to the shortest path between X_1 and X_2 in terms of neighborhood transitions (or local search steps or landscape edges). The larger the value $d(X_1, X_2)$, the more local search steps are needed to be able to reach X_2 from X_1. This correlation property is particularly important in a memetic context, e.g., without this correlation, the local search operator could reach very distant points in a few steps—this would minimize the relevance of the distance measure. We propose the following spacing indicators to express diversity:

- The *minimum spacing* $S(Pop) = \min\{d(I_i, I_j) : I_i, I_j \in Pop,\ i \neq j\}$;
- The *average spacing* $\overline{S}(Pop) = \frac{\sum_{1 \le i < j \le |Pop|} d(I_i, I_j)}{\frac{1}{2}|Pop| \cdot (|Pop|-1|)}$.

Given an element $X \in \Omega$ and a radius $R \in I\!R_+$, the R-sphere centered at X is denoted by $\mathscr{S}_R(X) = \{X' \in \Omega : d(X, X') < R\}$. The value of $S(Pop)$ indicates the maximum radius R such that $\forall I \in Pop$, $\mathscr{S}_R(I)$ does not include any element from $Pop - \{I\}$.

2.1.2 Rationale

Measuring and comparing diversity can be a delicate issue. Using only one spacing indicator is not always entirely satisfactory. For instance, $\overline{S}(Pop)$ might be reasonably high even if the population distribution is rather inadequate for a memetic algorithm, e.g., consider a few very distant clusters of close individuals. Regarding the minimum spacing, $S(Pop)$ can be 0 only because an individual is duplicated, even if the global population distribution is otherwise almost ideal. For these reasons, SMA pursues *two* spacing objectives:

Objective 1: Keep the minimum spacing $S(Pop)$ above a specific threshold R;

Objective 2: Subject to Objective 1, maximize the average spacing $\overline{S}(Pop)$ (as soon as $S(Pop) \ge R$).

Objective 1 aims at ensuring that there is no R-sphere $\mathscr{S}_R(I)$ (with $I \in Pop$) covering other Pop individuals except I. As soon as this property holds, SMA tries to *create* more diversity by pursuing Objective 2. Notice that Objective 1 can be expressed in terms of a *Maximum Minimum Diversity Problem* (MMDP) and Objective 2 as a *Maximum Diversity Problem* (MDP). These two problems are long-acknowledged in operations research and several specific MMDP and MDP algorithms have been developed since the 1990s [13].

To reach these objectives, the Spacing Memetic Algorithm (see Algorithm 1, or the complete source code[1]) introduces:

1. an offspring rejection procedure (Section 2.2);
2. a new replacement approach based on both spacing and quality (Section 2.3);
3. extensions and advanced techniques—reactive dispersion, implicit tunning of the mutation rate, fitness-spacing proportionate replacement, etc. (Section 2.4).

[1]Publicly available at `www.lgi2a.univ-artois.fr/~porumbel/sma/`.

Algorithm 1 SMA template

```
1: Input: a search space $\Omega$, a fitness function $f$ to minimize,
   a distance measure $d$, and a minimum spacing threshold $R$
2: Output: the best individual ever visited
3: Initialize parent population $Pop \leftarrow \{I_1, I_2, \ldots, I_{|Pop|}\}$
4: while a stopping condition is not met do
5:    $rejects \leftarrow 0$ {offspring rejection counter}
6:    repeat
7:       $parents \leftarrow$ SelectParents($Pop$)
8:       $O \leftarrow$ Recombination($parents$)
9:       $O \leftarrow$ LocalSearch($O$)
         {Offspring Rejection and Acceptance (§2.2)}
10:      $rejects \leftarrow rejects + 1$
11:      if $rejects > maxRejects$ then
12:         $O \leftarrow$ Mutation($O$)                {§2.2, §2.4}
13:         break
14:      end if
15:   until $d(O, I) \ge R\ \forall I \in Pop$
      {Standard SMA Replacement (§2.3)}
16:   if $d(I_i, I_j) \ge R\ \forall I_i, I_j \in Pop,\ i \neq j$ then
17:      $C_1 \leftarrow$ fitnessSelection($Pop$)     {problem-specific}
18:      $C_2 \leftarrow$ closestTo($C_1$)     {closest individual to $C_1$}
19:   else
20:      $\{C_1, C_2\} \leftarrow argmin_{\{I_i, I_j\}}\, d(I_i, I_j)$
21:   end if
22:   $Pop \leftarrow Pop \cup \{O\}$ - leastFit($C_1, C_2$)
23: end while
```

2.2 Offspring acceptance and rejection

Since the first objective of our diversity strategy is to maintain a minimum spacing threshold R, SMA strives to insert a new individual in the population only if its distance to each existing individual is greater than R. Consequently, if an offspring solution O is situated at less than R from an individual I, the standard behaviour is to "reject" O and to try to re-generate a new suitable offspring solution (see the `repeat-until` loop in Algorithm 1).

There are two exceptional cases that need to be addressed. First, if $d(I, O) < R$ but O is more fit than all individuals found so far, it would be quite unreasonable to reject O. For this case, we apply a direct replacement operation: insert O and directly eliminate I. This exception is similar to an "aspiration criterion" case in tabu search.

The second particular case that SMA needs to address is the risk of falling in an infinite loop by rejecting all new individuals (via the `repeat-until` loop). To overcome this risk, a maximum number of rejections per generation (denoted by *maxRejects*) is imposed. Only if this threshold is reached, SMA abandons creating diversity via the natural reproduction process (recombination and local search) and applies a mutation on O (Lines 12–13). One of the principles of our spacing policy is to ensure diversity without sacrificing quality, and so, SMA introduces "artificial" mutated individuals as rarely as possible, only as a last-resort tool. The mutation could consist of perturbing a certain number (*mutation strength*) of randomly chosen genes; see more advanced discussions on mutation use in Section 2.4.2.

2.3 Replacement based on spacing and quality

Besides pursuing diversity objectives, the replacement operator needs to follow at the same time the "survival of the fittest" principle. The idea of the standard SMA replacement (Lines 16–22, Algorithm 1) is to start out by first selecting two *close* candidates for elimination; only the least fit of them is eventually eliminated.

When SMA reaches the replacement stage (Line 16), the first spacing objective is usually ensured by the rejection procedure—except in certain special situations detailed below. As such, if $S(Pop) \geq R$, the replacement strategy only pursues the second of the spacing objectives: maximize the average spacing $\overline{S}(Pop)$. For this, it gets rid of small distances between existing individuals. The first elimination candidate C_1 can be chosen using any existing problem-specific selection, like roulette wheel or tournament selection (Line 17). The second candidate C_2 is chosen based on the following *spacing criterion*: C_2 is the closest individual to C_1, according to the given distance. SMA finally removes the least fit from C_1 and C_2—see Line 22.

An exceptional case might arise if the minimum spacing threshold R is not assured when SMA reaches the replacement stage. This situation can be due to mutations or direct replacements performed in the past. Such "anomaly" needs to be corrected by the replacement operator. As such, if $S(Pop) < R$, one selects as elimination candidates the *closest two* individuals (see the `Else` branch, Lines 19-20); the least fit is finally removed.

The above `Else` branch is the only SMA step that might require calculating $O(|Pop|^2)$ distances. Indeed, all other steps of Algorithm 1 (e.g., in Line 15 or 18) only require computing $O(|Pop|)$ distances. The test at Line 16 is not always necessary, e.g., the test is true unless previous mutations or direct replacements have been involved in the past. In such a case, the `Else` branch is no longer executed and the $O(|Pop|^2)$ distances are no longer calculated; more distance computations can further be saved using streamlining routines (by keeping a sorted table of distance values). In memetic algorithms, the population size $|Pop|$ is typically much smaller than the number of local search iterations (e.g., 10 compared to 100000 in Section 4.3). The conclusion is that the $O(|Pop|)$ distance computations induce a globally *limited slowdown* to the memetic process.

The proposed survival selection is very general and modular: it can directly re-utilize any existing problem-specific replacement, via the `fitnessSelection` routine—Line 17. In our experiments, this routine applies a fitness-proportionate selection: the probability to select individual I_i is $\frac{f(I_i)-f_{\text{norm}}}{\sum_{I\in Pop}(f(I)-f_{\text{norm}})}$, where f_{norm} is a normalization term (we used $f_{\text{norm}} = f_{\text{best}} - 1$, f_{best} being the best fitness). If one does not insist on re-utilizing the existing problem-specific survival, the *fitness-spacing proportional replacement* from Section 2.4.3 might constitute a more effective alternative.

2.4 Advanced spacing techniques and SMA options

The proposed template has certainly many possible variations and problem-specific knowledge can always enhance a (meta-)heuristic approach. For instance, an interesting option consists of applying the mutation before the local search so as to avoid introducing unfit individuals in the population. We discuss below three extensions that seemed to be among the most promising in our experimental context.

2.4.1 How to set the minimum spacing threshold R

A critical issue with SMA is to determine a suitable value of R. Such a value should ensure that any two individuals distanced by less than R share an important part of their genes. If R is too small, SMA risks to accept *too often* offspring bringing no new genetic material to the population. Ideally, R can be set using problem-specific search space motivations, e.g., by observing a clustering of high-quality individuals, by studying the size of the basins of attraction, plateaux, etc.

However, one can also set R dynamically during the search, by "reacting" on the number of offspring rejections per generation. In this approach, SMA could start with a very low R value (even 0) and gradually increase R as long as there is a small number of rejections per generation. The value of R can be considered "too large" from the moment when SMA can no longer maintain the minimum spacing above R without resorting to an extensive use of mutations. Recall that SMA can start using mutations only after $maxRejects$ rejections without any acceptance (see Section 2.2). An opposite approach (start with a large value and iteratively decrease it) proved successful in a different context, not depending on mutations [21, §4.4].

2.4.2 Reactive dispersion

While mutations can easily ensure diversity, they do not always compensate the associated quality loss. SMA follows the principle "diversity without quality sacrifices". As such, one should use a *high* $maxRejects$ value, so as to trigger a limited number of mutations throughout the search. However, SMA can allow mutations more easily when it detects certain stagnation situations, i.e., when the search process is blocked looping on deceptive search space structures ("traps"). In this case, the values of R and $maxRejects$ can be adapted via the "reactive dispersion" procedure described below.

Consider the following situation: the population converges toward a "stable" state with $S(Pop) \geq R$ but $\overline{S}(Pop) < 2R$; this can be due to certain particular search space "traps", e.g., numerous plateaus confined in a deep and large "well" that *attracts most local search processes.* To deal with such a situation, reactive dispersion is used. First, SMA increases the value of R, and so, more subsequent offspring solutions from this "well" will be rejected. By also reducing considerably $maxRejects$, the reactive dispersion leads to more frequent mutations, allowing the population to unlock itself from the "trap". The parameter changes are reversed as soon as $\overline{S}(Pop) \geq 2R$.

The same reactive dispersion can also be triggered on the following situation: if there are too many offspring rejections since the beginning of the search—i.e., if the average number of rejections per generation is always substantially higher than normal. This behaviour can arise if the population is distributed around several local optima with very strong basins of attractions, leading the natural reproduction to offspring in the same basins.

2.4.3 Fitness-spacing proportionate replacement

The standard SMA elimination scheme (Section 2.3) has the advantage of re-utilizing the problem-specific replacement operator (see Line 17). However, if one does not require this re-utilization, the following approach can be used for the case $S(Pop) \geq R$ (replacing Lines 17-18). First, each individual I_i is assigned a *fitness proportionate* probability $p(I_i)$. To select C_1, one picks up an individual I uniformly at random and performs one of the following: (i) with probability $p(I)$, set $C_1 = I$, or (ii) with probability $1 - p(I)$, pick up another random individual I and repeat with (i).[2] To choose C_2, one first picks up the closest individual I' to C_1 and performs a similar selection method: (i) with probability $p(I')$, set $C_2 = I'$, or (ii) with probability $1 - p(I')$, try with the next closest one and repeat. Except in extreme cases, this scheme requires computing only $O(|Pop|)$ distances.

3. RELATED WORK

3.1 Distance, spacing and SMA related ideas

Perhaps rather surprisingly at first glance, although distances are often used for various purposes in different contexts, *few* distance-based models of systematic diversity control are available in the literature. To our knowledge, there are only two important research directions showing direct similarities with SMA: (i) certain offspring addition criteria in combinatorial optimization (Section 3.1.1), and (ii) crowding techniques used for standard genetic algorithms in multimodal optimization (Section 3.1.2). Furthermore, many of these distance-related ideas have been developed in rather disparate research threads—e.g., the crowding techniques discussed in Section 3.1.2 are not regularly cited by papers from Section 3.1.1 or Section 3.2.

3.1.1 Distance-guided offspring addition tests

Memetic Algorithms with Population Management (MA|PM [20]) use an "input function" to decide if a new individual should be added to the population. This decision takes into account the solution quality and its distance to the population; any rejected offspring is directly mutated. The MA|PM "input function" shares certain goals with the offspring rejection from Section 2.2, but the novelty in SMA is that mutations are avoided. Indeed, SMA invokes mutations *much more rarely*, only as a *last-resort* diversification tool. Following the principle of "diversity without quality sacrifices", SMA avoids adding unfit individuals (which could be generated by using more mutations, less local search, or other means). However, the studies on MA|PM showed the effectiveness of the general idea of distance-based diversity control in several operations research problems—e.g., multidimensional knapsack or job scheduling. Useful ideas about policies to control R (i.e., the diversity parameter Δ in MA|PM) are presented in [20, §2.2.4].

Another interesting method of evaluating solutions for addition is used by the $(\mu+\lambda)$ evolutionary strategy extension from [21, §4.4]. All $\lambda + \mu$ (new and old) solutions are first sorted according to their cost; then, they are *iteratively* considered for addition in the new population (μ solutions need to be kept). At each iteration, the current solution (new or old) can be disregarded if its distance to any accepted solution is smaller than a threshold that is gradually lowered. More generally, a widely-used method of preserving diversity consists of removing duplicates from the population.

[2]If necessary, this operation might be repeated a few times before finally choosing an individual. Unless all p values are 0, the probability of never accepting an individual is 0.

3.1.2 Crowding in multi-modal optimization

Standard genetic algorithms can often be used to (try to) locate all global optima of a multi-modal function. In this context, each optima can be exploited by a sub-population that can be seen as a sub-species [15] specialized (crowded) on a "niche". One can even promote crossover only inside the subpopulations [15, 2] ("intra-niche" crossover), so as to "crowd" new individuals on the same niches. One of the best-known niching methods is crowding [4, 2]. This technique, popular in continuous optimisation, is commonly used to "induce niches by forcing new individuals to replace individuals that are similar genomically" [19]. For this purpose, the eliminated individual is selected from among the closest individuals to the offspring solution.

Such crowding elimination schemes share ideas with the SMA spacing-oriented elimination, but certain objectives are rather different. In crowding, the new individual replaces one from its own sub-population so as to induce population crowding on its niche; this way, it "preserves the diversity of the existing mixture" [15]. In SMA, stable subpopulations are generally *discouraged*: the rejection procedure actually tries to forbid individuals from crowding. Secondly, our memetic approach is based on a small-but-dynamic population that tries to continually create new diversity, and so, "inter-niche" crossover is preferred to "intra-niche" crossover. Finally, unlike in crowding, the SMA replacement does not take into account the current offspring in the elimination decision—the removed individual is not necessarily close to the new one.

3.2 Other connections to previous approaches

3.2.1 Distance-guided mating and recombination

In order to ensure that the offspring solution is always "different enough" from its parents, one can refer to special strategies of parent selection and recombination. This is particularly important in memetic algorithms, because mating close parents can easily lead the local search to very similar individuals. To avoid such issues, the parent selection can favor selecting *distant* parents (e.g., by avoiding incest [6]); the recombination operator can be designed so as to construct the offspring solution at equal distances from each parent. Such "distance preserving" recombination operators are quite common for certain problems, e.g., the distance-preserving crossover (DPX) for TSP [8].

However, SMA does not modify the problem-specific mating or recombination: Lines 7–8 in Algorithm 1 are the same as in a standard MA. For modularity reasons, SMA addresses all above issues *implicitly*, via the offspring rejection procedure.

3.2.2 Distantly related literature

Distances are often used for solution ranking in multi-objective optimization [5]. However, such diversity measures are typically calculated in the *objective function* space and they rely on fitness differences—not particularly meaningful in our mono-objective context. More generally, in scatter search and path-relinking, combinations typically construct offspring solutions by considering both the solution quality and its distance to its parent solutions [9].

Distance measures can also be used to detect premature convergence or genetic drift, and so, to trigger certain operators (e.g., restarts in the "CHC algorithm" [6], diversification operators in "diversity-guided evolutionary algorithms" [23], random immigrants, perturbations, partial restarts, etc.). In these research threads, distances measures are not actually required, but any *global* diversity indicator can be used.

4. EXPERIMENTAL RESULTS

This section is devoted to assessing the impact of the main spacing ideas within several case studies on different problems. For each problem, we provide comparisons between several algorithm versions, each with a specific component disabled. For instance, the algorithm version *Obj. 1 Off* represents a SMA variant in which the first spacing objective is not pursued (i.e., the minimum spacing threshold R is 0). MA represents a standard memetic algorithm in which no spacing techniques are used.

4.1 Artificial problems

4.1.1 One Min Plateau

SMA is first evaluated on the *One Min Plateau* problem, a variant of *One Max* extended with an additional artificial plateau. More precisely, this problem is defined in the search space of bit strings of length n and requires minimizing the objective function $f_{\text{plat}} : \{0,1\}^n \to \{0,1,\ldots,n\}$ defined by:

$$f_{\text{plat}}(X) = \begin{cases} ones(X) & \text{if } ones(X) < p_1 \\ p_1 & \text{if } p_1 \le ones(X) \le p_2 \\ ones(X) - (p_2 - p_1) & \text{if } ones(X) > p_2 \end{cases},$$

where $ones(X)$ represents the number of genes (bits) with value 1; p_1 and p_2 represent two "border" points of the artificial plateau of all individuals with a number of ones between p_1 and p_2. The value $d = p_2 - p_1 + 1$ can be interpreted as a plateau diameter; the total number of solutions inside the plateau is $\sum_{i=0}^{i=d-1} \binom{n}{p_1+i} = \sum_{i=0}^{i=d-1} \frac{n!}{(p_1+i)!\cdot(n-p_1-i)!}$.

Figure 1 depicts a projection of the landscape in a search space with 2 coordinates (x and y). The behaviour of a classical memetic algorithm consists of scattering the individuals in the plateau of solutions with fitness p_1 (this plateau attracts local search processes launched from lower quality solutions). The optimum can only be found when the crossover of two individuals leads to an offspring solution inside the valley of the global minimum. The results below show that this fortunate outcome occurs much more often when spacing techniques are used.

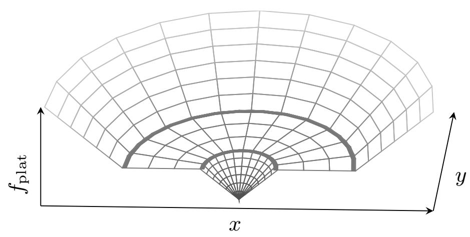

Figure 1: Projection of the *One Min Plateau* landscape, considering fitness function f_{plat} and the one-flip neighborhood. Without spacing strategies, classical memetic algorithms can easily get stuck in the plateau around the valley of the global minimum.

The well-known bit-flip neighborhood is used; the local search operator is a classical steepest descent algorithm without side-steps. The memetic scheme uses a classical Uniform Crossover [22], i.e., the offspring solution inherits all genes that are equal in the two parents and half of the non-matching genes from each of the two parents. The Hamming distance has been used; as such, one should be aware that the distance between two candidate solutions X_1 and X_2 represents the shortest path between X_1 and X_2 in the landscape associated with the bit-flip neighborhood.

All instances have $n = 100$ and we used several values of the plateau diameter $d = p_2 - p_1 + 1$, from 5 to 10 (p_2 is fixed to 25 and p_1 varies from 21 to 16). The parameters of the algorithms are: $R = 10\% \cdot n$, $maxRejects = 10$ and $|Pop| = 10$; the mutation strength is $10\% \cdot n$. The main performance indicator is the *success rate*: the proportion of runs finding the global optima out of 100 tries.

Generally speaking, the local search is the most time consuming operator in memetic algorithms. After each local search application, SMA typically performs at most $O(|Pop|)$ distance calculations (maximum $O(|Pop|^2)$ only if $S(Pop) < R$, see Section 2.3); furthermore, the (Hamming) distance calculation has linear complexity for our problem. Since a fixed number of local search iterations is systematically applied after each crossover, an acceptable machine-independent stopping condition is to reach a cut-off number of crossovers applications—i.e., we used 10000 for all SMA versions.

Plateau Diameter	SMA	MA	Obj. 1 OFF	Obj. 2 OFF
5	100/100	20/100	47/100	99/100
6	97/100	15/100	29/100	84/100
7	70/100	8/100	19/100	54/100
8	53/100	5/100	10/100	23/100
9	20/100	1/100	6/100	14/100
10	5/100	1/100	4/100	4/100

Table 1: Success rate comparison between SMA (Column 3) three other SMA versions on *One Min Plateau*. Several instances are considered; the plateau diameter ($d = p_2 - p_1 + 1$) varies from 5 to 10.

Table 1 presents a comparison between SMA (Column 2), the standard MA (Column 3), and two other SMA versions each with a specific component disabled—i.e., the first (and, respectively, the second) spacing objective is disabled in Column 4 (and, respectively, in Column 5). SMA attains clearly improved results compared to *MA*: the global optimum is reached between 5 and 20 times more often. The last two columns show that the standard MA can also be substantially improved only by pursuing separately one of the two proposed spacing objectives (the success rate becomes at least double).

4.1.2 NK Model

The second experiment on artificial problems concerns the well-known NK landscape model [12]. We used the same solution encoding, neighborhood, crossover, local search and parameters as for *One Min Plateau*. The source code is essentially the same, but we only defined a new objective function f_{NK} that requires maximization:

$$f_{NK}(X) = \sum_{i=1}^{n} g(X_{i+1}, X_{i+2}, \ldots, X_{i+K}),$$

K	SMA			*MA*			*Obj. 1 OFF*			*Obj. 2 OFF*		
	bst(#bst)	avg(std)	min	bst(#bst)	avg(std)	min	bst(#bst)	avg(std)	min	bst(#bst)	avg(std)	min
2	650 (2)	647.6 (1.2)	647	641 (3)	637.1 (3.0)	632	644 (1)	640.1 (1.9)	638	647 (8)	646.4 (1.2)	644
3	596 (1)	590.8 (2.6)	588	576 (1)	569.2 (4.0)	564	584 (1)	573.2 (6.5)	564	592 (8)	591.2 (1.6)	588
4	725 (7)	722.9 (3.2)	718	678 (1)	661.3 (14.1)	630	682 (1)	672.1 (8.3)	656	725 (4)	719.1 (5.7)	709
5	793 (5)	784.1 (8.9)	774	774 (1)	748.7 (16.5)	719	776 (2)	766.8 (9.3)	753	793 (4)	782.8 (8.3)	776
6	730 (3)	726.5 (2.3)	725	720 (1)	710.7 (6.4)	700	725 (2)	717.1 (7.2)	698	730 (6)	728.0 (2.4)	725
7	773 (1)	763.1 (7.8)	749	720 (1)	695.0 (12.1)	669	737 (1)	712.0 (17.4)	684	772 (2)	761.2 (8.3)	746
8	900 (1)	847.8 (25.3)	820	716 (1)	687.4 (17.0)	669	774 (1)	730.5 (27.2)	693	900 (1)	847.9 (21.7)	825
9	831 (1)	806.2 (15.0)	780	754 (1)	731.5 (15.0)	710	788 (1)	755.1 (17.6)	727	820 (1)	807.5 (8.9)	788
10	698 (1)	693.0 (3.0)	689	658 (1)	646.0 (8.4)	636	683 (1)	667.1 (9.3)	649	699 (1)	691.0 (4.7)	683
11	700 (2)	695.7 (4.3)	688	666 (1)	645.6 (12.8)	626	695 (1)	677.7 (6.0)	674	702 (2)	694.4 (5.1)	687

Table 2: Results of SMA (Column 2-4) and of three other SMA variants on ten *NK Model* problem instances (with K from 2 to 11 and $n = 100$). For each algorithm version, we report the best value ever reached (bst) in ten runs, the number of runs reaching this best value (#bst), the average result (avg), the standard deviation (std) and the worst fitness ever reached at the end of a run (min).

where all index additions are performed Modulo n. This objective function[3] defines a *NK Model* problem with adjacent (neighboring) lookup table index positions. We consider $n = 100$ and we generated ten random instances with K from 2 to 11 (the 2^K possible values of function g were randomly chosen from the set $\{0, 1, \ldots, 10\}$).

Tables 2 reports results on these instances, considering the same four SMA variants as for *One Min Plateau*. The performance indicator is here based on statistical measures over 10 runs—e.g., the best and the average solution quality reached at the end of all runs, as described by the legend of Table 2. Obviously, SMA can offer a significant improvement over *MA*: the worst solution reported by SMA is better than the best reported by *MA* (for all but one instance). The results of *Obj. 1 Off* and *Obj. 2 Off* provide an estimate of the individual impact of each of these two objectives.

4.2 The maximum clique problem

The SMA algorithms for the above artificial problems could also be applied to more practical combinatorial or numerical optimization problems. Indeed, our SMA implementation from the previous section (publicly available online[4]) only required the following two modifications to solve the well-known *Max Clique* problem: the objective function was redefined and new file input routines were written (to read inputs graphs). More precisely, given a graph $G = (\{v_1, v_2, \ldots, v_n\}, E)$, a bit string X of length n defines a *Max Clique* candidate solution in which a vertex v_i is selected if and only if $X_i = 1$. The objective function (we present here the maximization version[3]) was defined as follows:

$$f_{CLK}(X) = \begin{cases} -abs\_edges(X) & \text{if } abs\_edges(X) > 0 \\ ones(X) & \text{if } abs\_edges(X) = 0 \end{cases},$$

[3]Technically, our programs minimize the negation of this function (for interoperability issues within the source code).
[4]The source code is available at `www.lgi2a.univ-artois.fr/~porumbel/sma/`. We provide a C++ one-file solution: *clarity and generality* have a much higher priority than specific technical features (e.g., data structures for calculation streamlining are *not* used). The effort of switching to a different problem is minimal; fitness functions for *One Min Plateau*, *NK Model* and *Max Clique* are already provided.

where $abs\_edges(X)$ represents the number of pairs of vertices not linked by an edge in X (absent edges). The first case of this definition is only introduced to define a penalty for improper candidate solutions (that are not cliques); all elements from $\{0, 1\}^n$ are included in the search space.

Table 3 reports results on several (small) DIMACS instances [11], considering the same presentation format, algorithms and parameters as for *NK Model*—only the maximum number of crossover applications is lowered to 1000. We also use the same statistical indicators as for *NK Model*—i.e., for each instance and algorithm, Table 3 reports the best (bst), the average (avg) and the minimum clique size (min) reached in ten runs; see also the legend of Table 2 for more details (e.g., the #bst and std indicators).

SMA appears to be superior to *MA*, but one can no longer state that the worst performance of SMA outperforms the best performance of *MA*. However, the best solution reached by SMA was never replicated by other algorithms (except *Obj. 2 Off* for one graph). SMA reaches (with limited effort) the global optimum for two graphs: this is an interesting performance considering that a very generic implementation was used, with no particular clique features except the fitness function.

4.3 The graph k-coloring problem

Given a graph $G(V, E)$ and an integer $k > 0$, the graph k-coloring problem requires finding a conflict-free coloring (i.e., adjacent vertices colored with different colors) with at most k colors. While the above three problems use a bit-string encoding, a *k-coloring* candidate solution encodes a partition of V into k classes (colors). The fitness of a solution is given by the number of edges with both ends in the same class. A neighborhood transition consists of transferring a conflicting vertex from one class to another, a vertex being conflicting if it shares the same color class with one of its adjacent vertices. The partition distance [10] is correlated with this neighborhood: the distance between two solutions X_1 and X_2 is equivalent to the smallest possible number of neighborhood transitions needed to reach X_1 from X_2.

The aim of the k-coloring study is to assess the potential of SMA to get closer to state-of-the-art results; as such, we also make use of more advanced SMA techniques mentioned in Section 2.4. For instance, the reactive dispersion routine

Graph	opt	SMA			*MA*			*Obj. 1 OFF*			*Obj. 2 OFF*		
		bst(#b)	avg(std)	min	bst(#b)	avg(std)	min	bst(#b)	avg(std)	min	bst(#b)	avg(std)	min
C125.9.clq	34	34 (3)	33.1 (0.8)	31	32 (2)	29.9 (1.4)	28	32 (2)	31.0 (0.6)	30	33 (4)	32.2 (0.9)	30
brock200_2.clq	12	11 (2)	9.9 (0.7)	9	10 (1)	9.0 (0.4)	8	10 (3)	9.2 (0.6)	8	11 (2)	10.1 (0.5)	9
brock200_4.clq	17	16 (1)	14.4 (0.8)	13	15 (1)	13.3 (0.9)	12	14 (3)	13.1 (0.7)	12	15 (3)	14.3 (0.5)	14
gen200_p0.9_44.clq	44	40 (1)	36.3 (1.3)	35	37 (1)	33.7 (1.3)	32	36 (1)	34.7 (0.8)	33	39 (1)	36.2 (1.2)	34
gen200_p0.9_55.clq	55	55 (1)	40.2 (5.8)	35	43 (1)	36.8 (2.6)	34	46 (1)	36.4 (3.6)	33	53 (1)	41.8 (5.7)	37

Table 3: Comparison of SMA (Columns 3–5) with three other SMA versions on *Max Clique*. Columns 1 and 2 indicate the graph and the optimum solution; Columns 3–14 have the same meaning as in Table 2.

is applied in all SMA variants except *React. Off.* Furthermore, we use the fitness-spacing proportionate replacement (Section 2.4.3); the mutation is performed before the local search.

Memetic evolutionary algorithms represent a well-established coloring approach [7, 3, 14, 18]. Following ideas from this research thread, we use a classical Tabu Search local improvement operator and a crossover based on combining color classes. The main parameters are: $R = 10\%|V|$ (based on a clustering hypothesis [17]), $maxRejects = 50$ and $|Pop| = 10$; the mutation strength is $10\% \cdot |V|$, the local search chain has 100000 iterations and the stopping condition consists of finding a solution or of reaching a cut-off time limit of 2.5 hours (on a 2.50Ghz Xeon processor). We selected ten of the most challenging instances from the standard DIMACS graphs [11], using the lowest k for which a legal k-coloring has ever been reported in the literature.

Table 4 compares the results of SMA (Columns 2-3) with those of four other SMA variants. For each instance and for each algorithm, we provide the success rate (columns #hits) and the average time in minutes over the successful runs (columns $T[m]$). This comparison enables us to evaluate several SMA ideas in greater detail:

SMA The *complete SMA* reaches globally the best results (Columns 2–3). It systematically finds solutions for most of these difficult instances, a very good performance in the coloring literature. We observed that SMA shows no premature convergence (in terms of spacing) and that the success rates *can* be improved by allowing more time;

MA With all spacing objectives disabled, the classical memetic algorithm reaches poor results compared to any other SMA variant. This shows the practical impact of *both* spacing objectives (keep $S \geq R$ and maximize $\overline{S}$);

Obj. 1 OFF Without considering the first spacing objective (i.e., $R = 0$), the algorithm reaches significantly lower success rates than the complete SMA. In most cases, this is due to obvious premature convergence: we observed that the average spacing is usually close to zero at the end of failed runs;

Obj. 2 OFF By *not* pursuing the second diversity objective (i.e., using only fitness proportional replacement and ignoring spacing criteria if $S(Pop) > R$), SMA may fail to ensure a wide covering of the search space, and so, it *cannot be very robust*; globally, it finds fewer solutions than the complete SMA;

React OFF Without reactive dispersion (with fixed R and $maxRejects$), the algorithm is not able to unlock itself from special "traps" of the search space. In certain cases, this can make the difference between a success rate of 46/50 and 1/50. By reactive dispersion, SMA is able to detect and react on its own stagnation periods (see Section 2.4.2). We empirically noticed that the complete SMA can trigger up to 100 times more frequent mutations during these periods, and so, overcome stagnation situations that could otherwise keep the search indefinitely blocked.

In addition to the previous three case studies, this coloring analysis confirms the potential of SMA to reach some of the best-known bounds on a realistic problem. However, state-of-the-art results are often achieved through joint application and coordination of several other search operators and of different technical features as well—e.g., routines for streamlining the calculation *have* been used. Table 4 shows that the spacing mechanisms can become a key ingredient in making the algorithm reach difficult bounds—e.g., advanced spacing techniques (i.e., reactive dispersion) can help the search process to deal with more deceptive search space traps that could pose problems to other diversity techniques.

5. CONCLUSIONS AND OUTLOOK

The proposed Spacing Memetic Algorithm (SMA) is a general framework which commits itself to a systematic control of population diversity. Based on a distance metric defined on the search space, SMA models population diversity via two spacing objectives: keep the minimum spacing above a threshold R, and, subject to this, maximize the average spacing among the population individuals.

SMA uses a small-but-dynamic population that tries to avoid convergence and to continuously discover new promising search areas. With such goals in mind, we designed advanced strategies for *offspring rejection* and *spacing-guided survival selection*. Furthermore, SMA follows the principle "diversity without quality sacrifices", and so, it reduces the use of mutations to minimum: mutations are only used as a *last-resort* diversification tool. Additionally, SMA can integrate more advanced *reactive dispersion* strategies that exceptionally allow more frequent mutations when it is necessary, i.e., when the search needs to overcome stagnation.

All SMA components can be "attached" to existing memetic algorithms with no modification on the problem-specific operators, i.e., SMA can re-use (without change) the existing MA crossover, local search, parent and survival selection, etc. Consequently, most MA can be "upgraded" to SMA, provided that one can define a suitable distance measure—correlated to the neighborhood and to the landscape. The number of distance calculations can be kept in very reasonable limits.

Acknowledgments
We thank the anonymous referees for their remarks that helped us to improve the paper.

Instance: G, k	SMA		*MA*		*Obj. 1 OFF*		*Obj. 2 OFF*		*React OFF*	
	#hits	T[m]	#hits	T[m]	#hits	T[m]	#hits	T[m]	#hits	T[m]
dsjc500.1,12	50/50	2	15/50	31	21/50	25	49/50	6	50/50	2
dsjc500.9,126	44/50	60	26/50	36	30/50	41	42/50	46	42/50	54
dsjc1000.1,20	50/50	33	0/50	–	3/50	38	30/50	38	49/50	31
dsjc1000.5,83	37/50	95	10/50	65	26/50	82	15/50	117	37/50	99
dsjr500.1c,85	46/50	75	3/50	27	2/50	0	10/50	85	1/50	1
dsjr500.5,122	42/50	44	10/50	8	9/50	4	22/50	25	13/50	29
flat1000.76,82	42/50	100	9/50	80	33/50	86	12/50	109	42/50	95
le450.25c,25	47/50	56	3/50	6	2/50	80	23/50	53	29/50	54
r250.5,65	49/50	38	4/50	28	4/50	34	24/50	32	25/50	45
r1000.1c,98	43/50	33	31/50	19	33/50	29	37/50	37	45/50	32
Total #hits:	*450 hits/500*		*111 hits/500*		*163 hits/500*		*264 hits/500*		*333 hits/500*	

Table 4: Graph k-coloring results of SMA (Columns 2–3) and of four other SMA versions. We provide the success rate (columns "#hits") and the average time in minutes over successful runs (columns "T[m]"). For each graph, the chosen value of k represents the proven chromatic number or the best known upper bound.

6. REFERENCES

[1] E. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004.

[2] W. Cedeño and V. Vemuri. Analysis of speciation and niching in the multi-niche crowding GA. *Theoretical Computer Science*, 229(1):177–197, 1999.

[3] D. Costa, A. Hertz, and C. Dubuis. Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics*, 1(1):105–128, 1995.

[4] K. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan Ann Arbor, MI, USA, 1975.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[6] L. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. Rawlings et al., editors, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[7] C. Fleurent and J. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63(3):437–461, 1996.

[8] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 616–621, 1996.

[9] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–684, 2000.

[10] D. Gusfield. Partition-distance a problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82(3):159–164, 2002.

[11] D. Johnson and M. Trick. *Cliques, Coloring, and Satisfiability Second DIMACS Implementation Challenge*, volume 26 of *DIMACS series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, 1996.

[12] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of theoretical Biology*, 128(1):11–45, 1987.

[13] C. Kuo, F. Glover, and K. Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185, 1993.

[14] E. Malaguti, M. Monaci, and P. Toth. A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, 20(2):302, 2008.

[15] B. L. Miller and M. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 786–791, 1996.

[16] P. Moscato. Memetic algorithms: a short introduction. In D. Corne et al., editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, 1999.

[17] C. Porumbel, J. Hao, and P. Kuntz. A search space "cartography" for guiding graph coloring heuristics. *Computers & Operations Research*, 37:769–778, 2010.

[18] C. D. Porumbel, J.-K. Hao, and P. Kuntz. An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. *Computers & Operations Research*, 37:1822–1832, 2010.

[19] R. Smith, S. Forrest, and A. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1993.

[20] K. Sörensen and M. Sevaux. MA|PM: Memetic algorithms with population management. *Computers and Operations Research*, 33(5):1214–1225, 2006.

[21] T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539, 2006.

[22] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 2–9, 1989.

[23] R. K. Ursem. Diversity-guided evolutionary algorithms. In *PPSN VII*, volume 2439 of *LNCS*, pages 462–471. Springer, 2002.