Soft Comput (2013) 17:285–300
DOI 10.1007/s00500-012-0893-6

FOCUS

On user-centric memetic algorithms

Ana Reyes Badillo · Juan Jesús Ruiz ·
Carlos Cotta · Antonio J. Fernández-Leiva

Published online: 17 July 2012

Abstract Memetic algorithms (MAs) constitute a metaheuristic optimization paradigm [usually based on the synergistic combination of an evolutionary algorithm (EA) and trajectory-based optimization techniques] that systematically exploits the knowledge about the problem being solved and that has shown its efficacy to solve many combinatorial optimization problems. However, when the search depends heavily on human-expert's intuition, the task of managing the problem knowledge might be really difficult or even indefinable/impossible; the so-called interactive evolutionary computation (IEC) helps to mitigate this problem by enabling the human user to interact with an EA during the optimization process. Interactive MAs can be constructed as reactive models in which the MA continuously demands the intervention of the human user; this approach has the drawback that provokes fatigue to the user. This paper considers user-centric MAs, a more global perspective of interactive MAs since it hints possibilities for the system to be proactive rather than merely interactive, i.e., to anticipate some of the user behavior and/or exhibit some degree of creativity, and provides some guidelines for the design of two different models for user-centric MAs, namely reactive and proactive search-based schema. An experimental study over two complex NP-hard problems, namely the Traveling Salesman problem and a Gene Ordering Problem, shows that user-centric MAs are in general effective optimization methods although the proactive approach provides additional advantages.

A. R. Badillo · J. J. Ruiz · C. Cotta (✉) · A. J. Fernández-Leiva
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, Campus de Teatinos, Universidad de Málaga, 29071 Málaga, Spain
e-mail: ccottap@lcc.uma.es

A. J. Fernández-Leiva
e-mail: afdez@lcc.uma.es

Keywords Memetic algorithm · Interactive evolutionary computation · User-centric optimization · Combinatorial optimization

1 Introduction

The need of exploiting problem knowledge inside evolutionary algorithms (EAs), and metaheuristics in general, in order to both obtain solutions of better quality and accelerate the optimization process has been repeatedly highlighted (Hart and Belew 1991; Wolpert and Macready 1997; Culberson 1998; Davis 1991). A number of different ways to incorporate knowledge have been reported in the literature and one can find proposals such as the design of specific genetic operators, the definition of intelligent representations with inherent information on them, or the hybridization with another techniques, just to name a few, Puchinger and Raidl (2005) and Moscato and Cotta (2010). In this context, memetic algorithms (MAs) (Moscato 1999; Moscato and Cotta 2003; Moscato et al. 2004; Krasnogor and Smith 2005; Neri et al. 2012; Neri and Cotta 2012). In this context, memetic algorithms (MAs) (Moscato 1999; Moscato and Cotta 2003; Moscato et al. 2004; Krasnogor and Smith 2005; Neri et al. 2012; Neri and Cotta 2012) are probably one of the most successful proposals (in the sense of being effective optimization methods) to date (Hart et al. 2005).

However, both evolutionary and MAs have still evident limitations and there still exists one main complication that lies precisely in the difficulty to characterize the subjective interest through a certain mathematical expression or algorithm that can be optimized. This difficulty is generally common to those problems in which the search has to be conducted (directly or indirectly, completely or partially) in a psychological space. We speak thus about those problems in which the search has to be conducted on spaces comprising candidate solutions which are not easy to evaluate mathematically.

Within the framework of metaheuristics—and more specifically of evolutionary computing—the solution that has been proposed is the so-called interactive evolutionary computing (IEC). In a broad sense, IEC is an approach based on the optimization of a certain target system, using evolutionary computing and interacting with a human user; in other words, the user can influence the evolutionary process when this is being executed. Traditionally, this interaction was based on the subjective assessment of the solutions generated by the algorithm; in this line, see for instance, the seminal work of Dawkins (1986) as well as different applications in artistic fields (see, e.g., the proceedings of EvoMUSART), industrial processing of audiovisual information, data mining or robotics, among other fields (Takagi 2001). The common nexus of classical IEC is the existence of a reactive search-based mechanism in which the user provides some feedback to the demands of the running EA.

Although IEC represents an extension to EC that makes it useful on problems that demands knowledge provided by human user, it is also true that classical IEC methods have still an important limitation (that is also inherent to the IEC model): the fatigue of the human user that is produced by the continuous feedback that the subjacent EC technique demands to the user. Advanced IEC techniques smooth this drawback by extending its concept to an optimization centered in the user in the sense that the interactive optimization process tries to guess the further user interactions and thus reduce the requirement of user interventions. This form of optimization has been termed as user-centric evolutionary computation (Parmee and Abraham 2004; Parmee et al. 2008) [note that the term "*Human-centric*" has also been used instead of "user-centric" (Parmee 2007)].

There still remains an important issue to analyze: the combination of user-centric evolutionary optimization and MAs. As already mentioned, perhaps the most prominent characteristic of a MA is the systematic exploitation of knowledge about the problem being solved, and IEC represents another form of incorporating knowledge to the problem. Therefore it seems natural to investigate a global combination of both components, termed here as user-centric MAs, with the goal of providing an extra dimension to each of their constituent parts. Actually, some works have already highlighted the benefits attainable via the use of the human user interaction with the MA, in particular in the context of multi-objective optimization (Dias et al. 2008; Jaszkiewicz 2004). We explore here some of these capabilities (extended to proactive models) in this work. In particular, this paper provides a general overview on user-centric memetic computation, providing principles for their design, identifying the places where a human user can interact with the subjacent MA under the demand of this algorithm (i.e., a reactive approach), and drawing a more general schema for a proactive model. Two study cases for the optimization of the Traveling Salesman Problem (TSP) and a Gene ordering problem (GOP) are also analyzed to show the adequacy of human-guided MA-based optimization.

2 Memetic algorithms

The adjective 'memetic' comes from the term 'meme', coined by Dawkins (1976) to denote an analogous to the *gene* in the context of cultural evolution. As EAs, MAs are also population based metaheuristics. The main difference is that the components of the population are active entities that cooperate and compete in order to find improved solutions. rather than mere passive solutions.

There are many possible ways to implement MAs. The most common implementation consists of combining an EA with a procedure to perform local search (LS) that is usually done after evaluation, although it must be noted however that the integration does not simply reduce itself to this particular scheme. In fact, the purpose of using LS inside a MA is to provide specific knowledge that can help to a better optimization process (Bonissone et al. 2006). For instance, Fig. 1 shows the classical view of a MA and

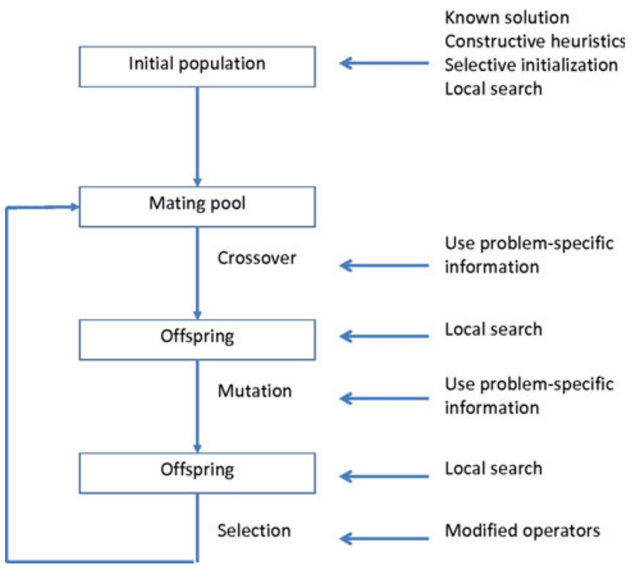


Fig. 1 Places to incorporate problem knowledge within an evolutionary algorithm, according to Eiben and Smith (2003)

indicates places were problem specific knowledge, in form of a local searcher, can be incorporated inside a specific metaheuristic (i.e., a genetic algorithm) according to Eiben and Smith (2003). Also, Algorithm 1 shows a general picture where a LS can be incorporated inside an MA (note that this classical combination follows an integrative approach as considered in Puchinger and Raidl (2005).

This classical schema considers partial Lamarckianism (Houck et al. 1997), in which the application of LS depends on certain probability p_{LS} so that LS might be applied only to a fraction of individuals (the individuals to which LS will be applied can be selected in many different ways (Nguyen et al. 2007). Note that applying always the LS in each generation of the MA (or initially on each individual in the initial population) is not always the best option (as shown in Sudholt 2009, for the application of LS on each generated new individual).

Algorithm 1: Pseudocode of a basic MA based on a integrative collaboration with a local search (LS) technique

```
1  for $i \in \{1, \ldots, \text{POPULATION SIZE}\}$ do
2      $pop[i] \leftarrow$ RANDOM-SOLUTION();
3      if Rand$[0,1] < p_{LS}$ then // LS is applied with
           probability $p_{LS}$
4          LS($pop[i]$);
5      end if
6  end for
7  $i \leftarrow 0$;
8  while $i < MaxEvals$ do
9      RANK-POPULATION ($pop$); // sort population
           according to fitness
10     $parent_1 \leftarrow$ SELECT ($pop$);
11     if Rand$[0,1] < p_X$ then // recombination is done
12         $parent_2 \leftarrow$ SELECT ($pop$);
13         $child \leftarrow$ RECOMBINE ($parent_1, parent_2$);
14     else
15         $child \leftarrow parent_1$;
16     end if
17     $child \leftarrow$ MUTATE ($child, p_M$); // $p_M$ is the
           mutation probability per gene
18     if Rand$[0,1] < p'_{LS}$ then // LS is applied
19         LS ($child$);
20     end if
21     $pop[\mu] \leftarrow child$; // replace worst
22 end while
23 return best solution in $pop$;
```

In general, the underlying idea of this kind of integration is to combine the intensifying capabilities of the embedded LS method, with the diversifying features of MA, i.e., the population will spread over the search space providing starting points for a deeper (probably local) exploration. As generations go by, promising regions will start to be spotted, and the search will concentrate on them. Ideally, this combination should be synergistic, providing better results that either the MA or the LS by themselves.

Regarding this issue, one can find in the literature a number of proposals that explore the intensification/diversification balance within the MA. Some works lean towards a more explorative combination, by using a blind recombination operator in the MA whereas other models incorporate an intense exploration of the dynastic potential (i.e., set of possible children) of the solutions being recombined (Cotta and Troya 2003; Gallardo et al. 2007).

In addition to other domains, MAs have proven to be very successful across a wide range of combinatorial optimization problems, where they are state-of-the-art approaches for many problems. For a comprehensive bibliography, the reader may consult Neri et al. (2012), Neri and Cotta (2012) and Moscato and Cotta (2007).

3 Why human-guided memetic algorithms?

EAs require that the user defines, before the process of evolution, the fitness measure (i.e., the evaluation function) that will be used to guide the evolution of candidate solutions. Those problems in which the fitness function is difficult (or even impossible) to formulate can hardly be handled by classical EAs; in this context IEC has recently been proposed as a part of evolutionary computation (EC) to cope with those problems that possess aesthetical or psychological features and as a consequence fitness evaluation functions are difficult, or even impossible, to formulate mathematically.

Generally speaking, IEC (also termed indistinctly here as user-centric EC or human-guided EC) represents an optimization paradigm that promotes the communication between a human user and an automated EA. The human usually intervenes under the demand of the subjacent EA, for instance to provide subjective fitness evaluation of candidate solutions. The classical version of IEC basically consists of incorporating human user evaluation during the evolutionary procedure.

In any case, more modern models of IEC have been proposed to attain the collaboration between the human user and the EA. For instance, Takagi (2000) proposes using techniques of dimensionality reduction to project the population of the EA to a bidimensional plane that is displayed to the user and over which the user selects the most promising candidates. It is also worthwhile to mention the work conducted in the area of multi-objective IEC (Deb and Chaudhuri 2007; Deb and Kumar 2007) in which the aim is to direct the exploration toward particular regions of the Pareto front. Again this kind of participation only represents one of the manifold forms that exist to fix search priorities. Interactive EAs have already been implemented in all the standard types of EC (as for instance in genetic programming (Lim et al. 2004; Lim and Cho 2005), genetic algorithms (Kosorukoff 2001), evolution strategies (Breukelaar et al. 2006), and evolutionary programming (Kubota et al. 2003) just to name a few. Interactivity has also been added to a number of cooperatives models (e.g., Babbar and Minsker 2006; Quiroz et al. 2008, 2009).

A recent work (Cotta and Fernández-Leiva 2011) describes the basic fundament of IEC, presents some guidelines to the design of interactive EAs to handle combinatorial optimization problems, and discusses the two main models over which IEC is constructed, namely reactive and proactive search-based schemas. In the reactive model the subjacent algorithm demands the direct intervention of the user whereas in the proactive model the subjacent algorithm constructs a model of the user's preferences that takes the role of the human user in the reactive schema. The objective of the proactive model is to mitigate the main problem of the reactive model, that is to say, the fatigue/tiredness that the human user accumulates as result of being continuously demanded from the underlying algorithm. Such fatigue can take different forms. One is the exhaustion after hours of work. However, even in shorter periods of time, a user subject to a repetitive task can inadvertently reduce his effort by paying less attention or providing less careful feedback.

In general IEC and human-guided search have been widely studied (see for instance Takagi 2001 and Klau et al. 2010) that present surveys respectively on these mentioned issues). However, no general approach for the design of effective interactive MAs exists in a well-defined sense, and hence this design phase must be addressed from an intuitive point of view as well. Recently, in Espinar et al. (2012) we have provided a first approximation to this issue and have formulated some principles for the design of reactive hybrid EAs; in this mentioned paper we also described a reactive MA for the search of optimal Golomb rulers, a very hard to solve combinatorial problem. Now, here we analyze interactive MAs from a more general perspective, discussing the principles for human-guided MAs, including both reactive and proactive interactive MAs. The aim is to help the reader to understand the mechanisms of human-guided MAs and provide some indications for their design.

4 Human-guided reactive MAs

In Espinar et al. (2012), we provided the first attempt (to the best of our knowledge) of establishing a global approach for the design of effective interactive MAs and defined a general schema for constructing reactive interactive MAs in which the human user interacts with the subjacent MA when the automated algorithm demands her intervention. In general, the human user might interact with the MA in a number of ways (the reader is referred to Espinar et al. 2012 for a more comprehensive explanation of these ways). This schema corresponded to a reactive model in which, from a global perspective, the basic idea is to let the user affect the search dynamics with the objective of driving (resp. deviating) the search towards (resp. from) specific regions of the solution space. The intervention of the user might be required asynchronously (e.g., the MA demands the user intervention because the search does not progress adequately and needs assistance from the human user), or synchronously (for instance by imposing a fixed number of human interventions). Of course, the human user might also act as a mere supervisor of the search process so that her intervention might be voluntary in any moment. In the following we analyze a study case in the context of the well-known Traveling Salesman problem (TSP).

4.1 A study case: a user-centric approach to MAs for TSP

As already mentioned in Sect. 1, MAs are particularly suited to integrate different sources of problem-knowledge into a single optimization tool. We refer to Moscato and Cotta (2010) for an up-to-date review of the state-of-the-art in MAs. In the following, we shall describe how we have integrated user-centric capabilities in MAs. In particular, we focus in the dynamic management of user-defined constraints, and in user-controlled LS.

4.1.1 Rationale

Some of the most common themes in IEC are using a human-expert to provide subjective evaluation information, or to perform subjective selection of solutions for breeding, among many others. We defer to Takagi (2001) for an overview of the area. One of the recurring issues in this context is dealing with human fatigue, i.e., coping with the fact that the human expert cannot be forced to provide a continuous supply of information, and hence the search algorithm has to exhibit a degree of autonomy. This is particularly feasible in domains in which some objective optimization measure is already available, and therefore the human expert is a source on knowledge that can improve results, but is not necessarily required for obtaining some solutions (even if just low-quality ones). In this sense, we adhere to this vision of having an human expert overseeing the evolution of resolution process, and providing hints (Abu-Mostafa 1993) on which directions the search should proceed but only sporadically (and asynchronously if possible).

More precisely, we have considered three particular ways to put the user in the loop, biasing the search dynamics:

- Allowing her to change dynamically some parameters of the algorithm, including the application probability and choice of operators (in order to change the way solutions are generated and thus direct the exploration process). Note in this sense that there are many works focusing in self-parameterization of EAs (Smith 2008). Thus, the human expert would here act as a high-level controller that would exert direct control of these parameters, or supervise the procedure of self-adaptation, superseding the latter if necessary.
- Allowing her to provide search bias via the dynamic introduction (and removal) of additional constraints, i.e., constraints that are not a part of the problem definition, but are forced by the user in order to drive the search towards-to/away-from specific regions of solution space. Such constraints are handled as soft-constraints, i.e., their violation results in a penalty term being added to the raw fitness of solutions.
- Allowing her to selectively use local-search add-ons. This is particularly relevant in the case of MAs, in which several studies exist focusing on which solutions should undergo local improvement, and how this local improvement should be done (i.e., which LS operator to use, how intense this local improvement has to be, etc.)—e.g., see Ong and Keane (2004) and Ong et al. (2006). Allowing the user to interfere in this regard allows further possibilities such as applying local-improvement just to particular portions of solutions rather than undergoing a full-fledged local optimization.

Next section will describe how we have accommodated the above capabilities in a memetic solver for the Traveling Salesman Problem (TSP).

4.1.2 *Implementation and management of user input: the TSP case*

We have built a prototype of user-centric reactive MA on the basis of the ECJ library.[1] ECJ is an evolutionary computation framework written in java available under the Academic Free License (AFL) version 3.0, and it has been chosen due to its high flexibility and modularity among other reasons. Our implementation comprises problem-specific classes (corresponding to the representation of solutions and variation operators used) and interaction-specific classes (providing the functionality for supplying information to the user and accepting feedback from her). Among the latter we can cite:

- `Output`: this class has been modified in order to allow the user select specific actions, e.g., modify parameters, introduce constraints, etc.
- `VectorSpecies`: a derived class `PermutationVectorSpecies` has been defined for the TSP in order to store problem-specific parameters and dynamic constraints.
- `Statistics`: a derived class from the former is responsible for controlling when user interaction takes place. In this prototype we have opted for two interaction possibilities: a pre-scheduled mechanism (interacting every certain number of generations; this is dynamically reconfigurable by the user, who can effectively set up when the next interaction will take place), and a trigger mechanism (interacting when the algorithms fulfills some condition, i.e., diversity drops below a certain threshold).
- `Canvas`: several problem-specific classes are derived from the latter in order to provide the means to display sensible information to the user.

The latter aspect is particularly important if the interaction with the user is to be fruitful. The user needs being provided with relevant (yet not overwhelming) information upon which to base her decisions on the course the search has to take. In this sense, the TSP has been chosen as test-suite precisely because of its amenability for graphical depiction, and intuitive visual nature. Figure 2 shows the basic interface. The left panel provides a description of the population: a graph is built by merging all tours in the population, subsequently, it is drawn making edge-width be proportional to the frequency of that edge in the population. As to the right panel, it provides a description of the best solution found and its quality. At the bottom, a drop-down menu provides the user a list of available actions (some of which can in turn result in additional lists of options and/or text inputs). An important feature is the possibility of selectively applying local-improvement to a specific portion of a solution. This is shown in Fig. 3. As it can be seen, the user can select a subset of the solution upon which 2-opt LS will be applied (i.e., only edges adjacent to selected cities can be modified). From a general point of view, this feature is important in order to make a better use of the computational effort (consider that LS consumes a large part of the computational budget of a MA) by focusing on specific portions of the solution that can benefit most of the application of LS, rather than blindly exploring the whole neighborhood of the solution. Obviously, this relies on the capability of the user to detect this issue which in turn is influenced by the particular problem considered, the visualization method used, and the size of the problem instance at hand. As the latter grows larger there may appear difficulties in conveying the

[1] http://www.cs.gmu.edu/~eclab/projects/ecj/.

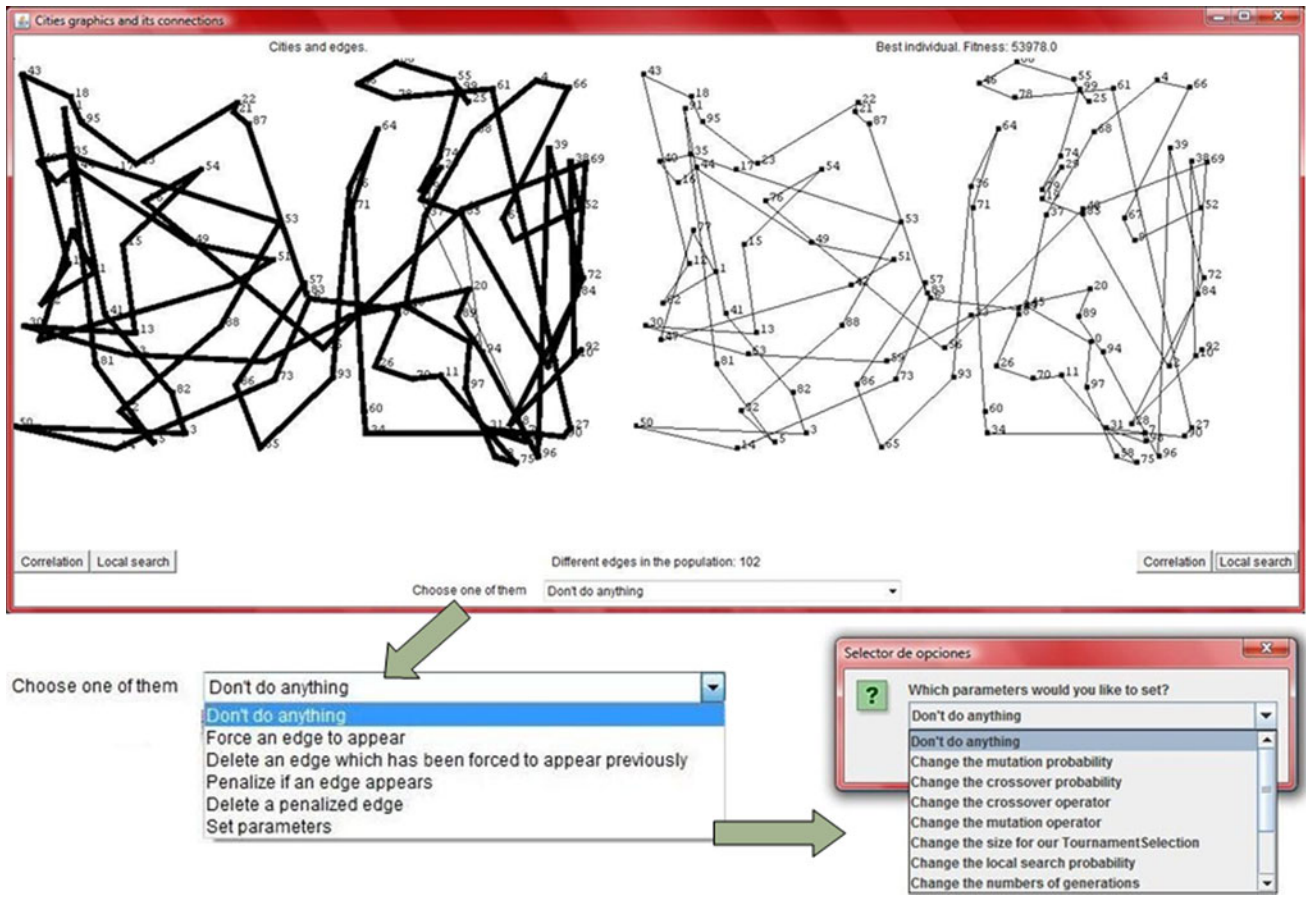

Cities graphics and its connections
Cities and edges.
Best individual. Fitness: 53978.0
Correlation
Local search
Different edges in the population: 102
Choose one of them
Don't do anything
Choose one of them
Don't do anything
Force an edge to appear
Delete an edge which has been forced to appear previously
Penalize if an edge appears
Delete a penalized edge
Set parameters
Selector de opciones
Which parameters would you like to set?
Don't do anything
Change the mutation probability
Change the crossover probability
Change the crossover operator
Change the mutation operator
Change the size for our TournamentSelection
Change the local search probability
Change the numbers of generations

Fig. 2 General depiction of the user interface for interacting with the memetic solver in the context of the TSP

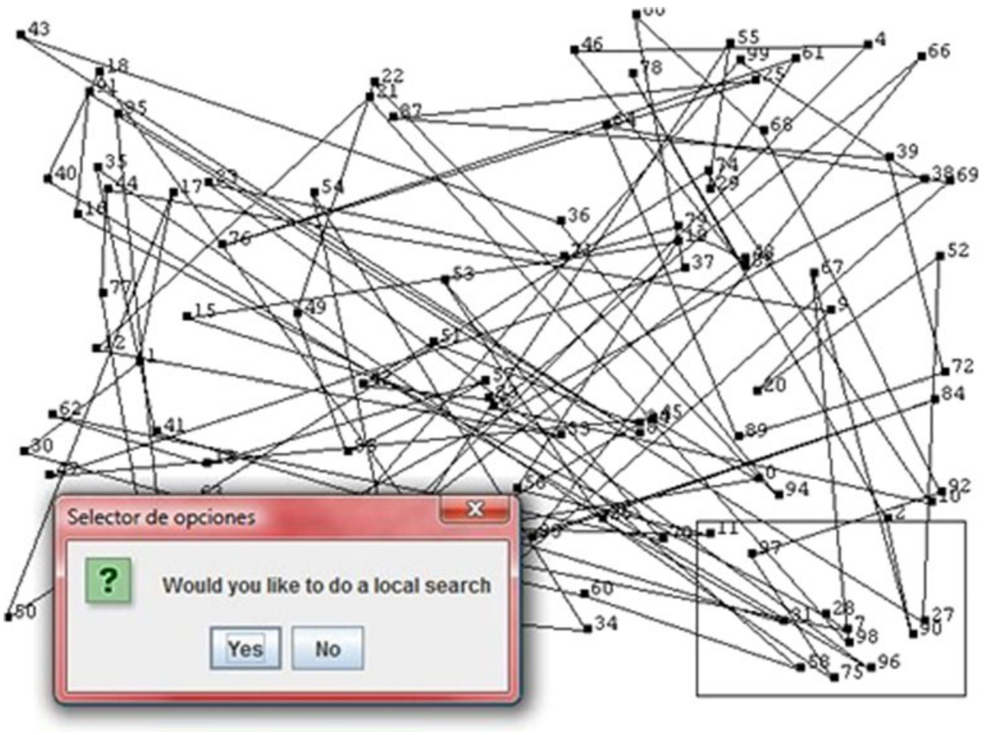


Fig. 3 The user can control the application of local search to specific portions of the current best solution

Table 1 User interaction in the `kroA100` instance

No. of interactions	Action performed
1	forbid $\langle 15-50\rangle$, $\langle 25-65\rangle$, $\langle 4-72\rangle$ and $\langle 43-68\rangle$
2	forbid $\langle 43-79\rangle$, $\langle 14-89\rangle$ and $\langle 62-73\rangle$
	2-opt LS in the bottom right corner
4	forbid $\langle 65-98\rangle$, $\langle 50-56\rangle$ and $\langle 50-60\rangle$
	forbid $\langle 21-82\rangle$, $\langle 22-68\rangle$ and $\langle 22-48\rangle$
	forbid $\langle 13-50\rangle$, $\langle 64-82\rangle$ and 2-opt LS in the bottom right corner
	forbid $\langle 57-62\rangle$ and 2-opt LS in the top left corner
8	forbid $\langle 14-30\rangle$, $\langle 13-46\rangle$ and $\langle 18-61\rangle$
	forbid $\langle 3-50\rangle$ and $\langle 43-54\rangle$
	forbid $\langle 23-71\rangle$ and $\langle 55-71\rangle$
	forbid $\langle 17-47\rangle$
	2-opt LS in the bottom right corner
	2-opt LS in the top left corner
	2-opt LS in the top right corner
	2-opt LS in the bottom left corner

information to the user. Then again, this is more an issue of data visualization –an interesting a substantial topic by itself– rather than an issue of the search algorithm.

4.1.3 Experiments

The experiments have been done using an elitist steady-state EA ($popsize = 100$, $maxevals = 10,000$, binary tournament selection) with edge-recombination crossover ($p_X = 1.0$), and subtour-inversion mutation ($p_M = 0.005$). Two TSP instances from the TSPLIB,[2] namely `kroA100` and `kroA200` have been used. In order to obtain baseline results, 20 runs of the algorithm have been done without user interaction. Subsequently, we have done single runs with 1, 2, 4 and 8 user-interactions. These interactions have been logged (specific actions and time at which they are done), and are subsequently replicated in automatic runs of the algorithm in order to determine their general goodness. Table 1 shows an example of the kind of actions performed on the `kroA100` instance. Six different users participated in these experiments.

The results are shown in Fig. 4. Notice how in the case of the `kroA100` instance the results are better for an increasing number of iterations, mostly due to the selective application of LS (which is much less expensive than a full-fledged LS, and whose cost is already accounted in the total computational budget). In the case of the `kroA200` such improvement is only attained for a larger number of interactions (which is where LS is effectively deployed). Except in `kroA100` and 1 interaction, in all cases the differences with respect to the autonomous algorithm are statistically significant at 5 % level using a Wilcoxon ranksum test.

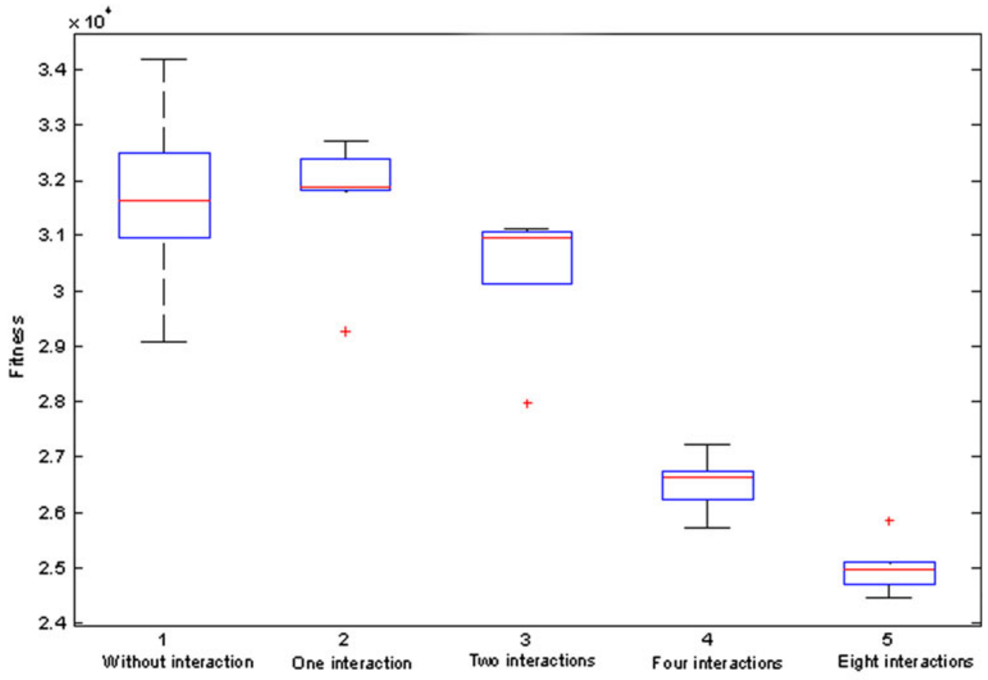
[2] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

5 User-centric proactive MAs

One of the drawbacks that can be observed in the use of user-centric reactive MAs (also termed here as reactive interactive MAs) corresponds exactly with one of the main concerns of classical IEC, that is to say, the fatigue of the human user that appears when reactive IEC algorithms are employed. The fatigue that an interactive MA causes in the human user is, as in an interactive EA, the result of demanding continuously feedback to the user. Several mechanisms described in the literature have been proposed in the literature to mitigate this fatigue in the context of IEC and these can be naturally extrapolated to the context of interactive MAs (Ohsaki et al. 1998; Sáez et al. 2005).

One proposal that mitigates this problem consists of replacing the reactive answer of the user by a proactive approach in which the subjacent running algorithm usually infers the user's answer before the feedback demand. In other words, during the interactive optimization, the underlying EA (in this case a MA) works to construct a model of the user's preferences; the objective is to reduce the number of user interventions by guessing her actions in those cases in which it would be necessary to demand her intervention (in these cases the user demand is replaced by the automated application of the guessed actions). This user model can be viewed as a prediction model and as a consequence might be constructed using computational learning techniques. This is a sophisticated approach in which the intervention of the user is optional and the algorithm runs autonomously (Breukelaar et al. 2006).

Proactive algorithms are not new and one can find a number of proposals in the literature; for instance Beck and Wilson proposed a set of proactive algorithms for the job shop scheduling problem with probabilistic durations (Beck and Wilson 2005, 2007); also, an ant colony optimization-based routing approach that proactively set up multiple paths between the source and the destination in a Mobile ad hoc network was described in Mamoun (2010). We can also mention other works such as Khanna et al. (2008). Moreover, an illustrative example of proactive algorithms might be the *Estimation of Distribution Algorithms* (EDAs) (Lozano et al. 2006) that were proposed by Mühlenbein and Paaß (1996) and departed from traditional EAs in that the generation of new solutions depended on a probabilistic mechanism, rather than on the use of a set of genetic operators. Relationships and dependencies among the variables that define a solution to the problem under consideration are explicitly expressed in EDAs via probability distributions.



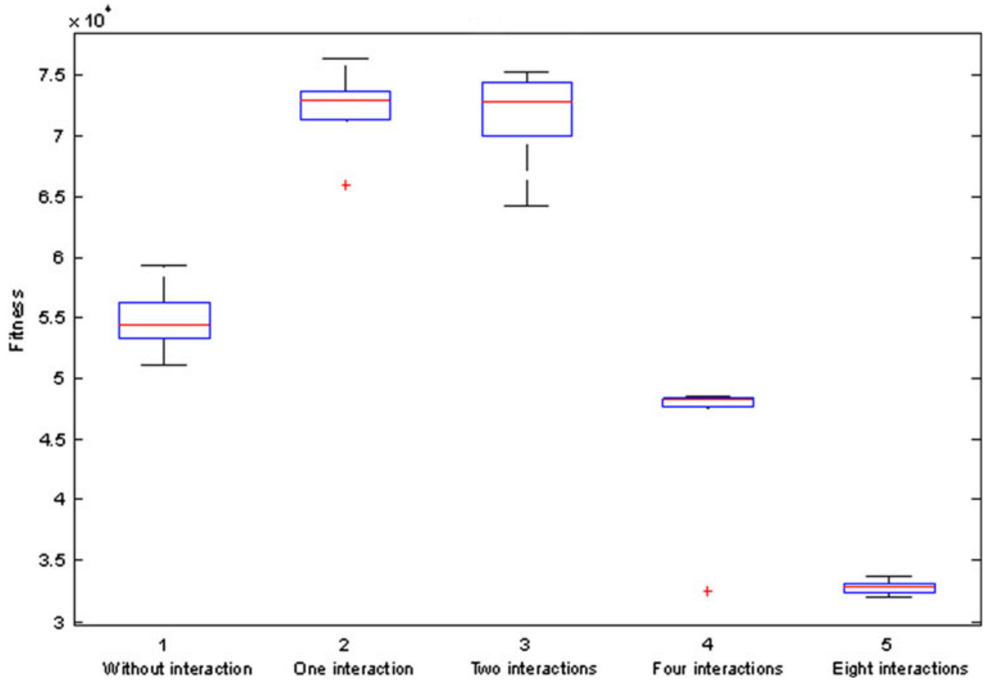


Fig. 4 Results obtained by interactive and non-interactive algorithms on the `kroA100` instance (*top*) and on the `kroA200` instance (*bottom*)

Generally speaking, (traditional) EDAs work as follows: an initial probabilistic model is built, typically describing a uniform distribution over the search space (although some heuristic initialization can also be done if problem-knowledge is available). This model is subsequently sampled to obtain a population of solutions from which an elite sample will be extracted and used to rebuild the model. The new model is then resampled and the whole process is repeated until a certain termination condition is fulfilled. This continuous process of updating and adjusting to the new conditions can be considered as an adaptive model in the sense of a proactive schema. However, proactivity, as proposed in these works, is not directly related with human interactivity but with approaches that can predict certain information that surely will be useful in the future and thus these algorithms can manage this information for their own convenience (for instance to tune themselves). In the context of pure IEC we can mention a number of works following this line of research such as Babbar and Minsker (2006), Gong et al. (2009), Inoue et al. (1999) and Dozier (2001); however, there are no memetic versions except the already mentioned work described in Espinar et al. (2012).

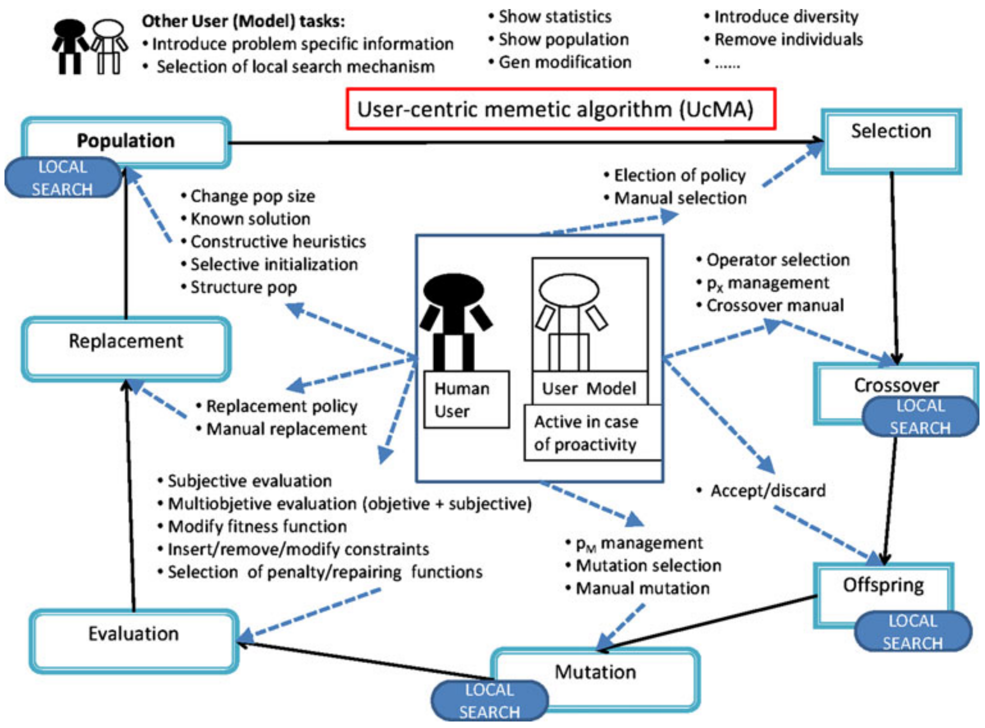


Fig. 5 A possible schema of a user-centric memetic algorithm

This section is devoted to present a *proactive user-centric memetic search/optimization*. This is the case when the interactive MA model employs computational learning techniques to predict the adequacy of the solutions still to be evaluated. Figure 5 shows a possible schema for a user-centric MA in which both the human user and the user model (i.e., the predictive model) can interact with the subjacent MA in a number of several forms. This schema extends the schema suggested in Espinar et al. (2012) in a number of ways that can be enumerated as follows:

- It introduces the role of a predictive model that will replace the human user to reduce (or even avoid the appearance of) her tiredness;
- It centers the interaction process in the user;
- It indicates new ways in which the user can influence the optimization process that were not mentioned in Espinar et al. (2012).

Regarding the latter issue, in general the human user (and the user prediction model) can influence the optimization process in several ways that basically coincide with those that were reported in Espinar et al. (2012) and that can be summarized as follows:

- The user might select the policy of the genetic operators (and the genetic operators themselves). She also might modify all the parameters of the algorithm (e.g., operators application probabilities).
- The user may influence the candidate population by ranking it according to some (possibly psychological) criteria (this of course has influence in the further replacement process), removing individuals, modifying individual representation (even at gene level), or even introducing/imposing certain level or criteria of diversity, just to name a few actions.
- The user can control the application of LS (if we consider the most classical form of a MA) in several levels. For instance, the user might decide to establish a partial Lamarckianism schema as mentioned in Sect. 2.
- The user might act as the evaluation mechanism. In this context the user might add the subjective evaluation as an additional component to the objective evaluation (for instance as an addend with some associated weight) or use the subjective value and the objective values as two different objectives to optimize (transforming thus the model in an interactive multiobjective MA). The user might also reformulate the objective function (or even add new objectives) and also with respect to the problem constraints.

It should be noticeable that the user is required to have certain knowledge about both the problem domain and the search process in order to obtain an effective interactive MA. In the latter case, the user needs certain level of expertise to cope with the optimization process from an algorithmic point of view so that she can manage the parameters that influence the search. In the first case, the user might use this knowledge for instance to assess candidates (or even mark the best/worst solutions), provide subjective information to the search if necessary, or control the use of local improvement by identifying the adequate regions of the search space to apply it.

Indeed, we might even think of having two different users working (perhaps in parallel) in these two distinct (but complementary) levels of knowledge.

If the prediction model of this adequacy is sufficiently adjusted, then alternating phases between optimization via the interactive MA and optimization via the predictive model can be conducted. This is precisely the idea that is shown in Algorithm 2. The general process basically works as follows: initially the MA runs autonomously and demands the attention of the human user when it detects that the search is not progressing adequately. Then the human user intervenes in the optimization process if her level of fatigue is acceptable. During this interactive process the automated algorithm constructs a prediction model of the user preferences that will be used in subsequent phases of the execution of the proactive user-centric algorithm. If the automated algorithm detects that the human user might be tired then the user (prediction) model is activated and substitutes the human user in the optimization process; this means that the human user will not be demanded by the MA what it is translated in a progressive reduction of her fatigue and, in case of search stagnation, the MA will impose the preferences proposed by the predictive model. Of course, the human user can always intervene voluntarily in the process; in any case, as the optimization continues it is expected that the tiredness of the human user progressively decreases so that the process can go back to the initial phase of interaction.

Algorithm 2: Pseudocode of a basic user-centric MA

```
1  INITIATE EXECUTION OF MA;
2  INITIALIZE USER MODEL $M_u$ AS EMPTY;
3  while not (stop criteria reached) do
4      if fatigue(user) < threshold then
5          if user intervenes then
6              ADD USER PREFERENCE ($U_1$) TO MA;
7              ADD $U_1$ TO USER MODEL $M_u$;
8              INCREASE FATIGUE VALUE;
9          else
10             DECREASE FATIGUE VALUE;
11         end if
12     else
13         if search does not progress then
14             GUESS USER PREFERENCE $U_2$ FROM USER
               MODEL $M_u$;
15             ADD USER PREFERENCE ($U_2$) TO MA;
16         end if
17         DECREASE FATIGUE VALUE;
18     end if
19     CONTINUE EXECUTING MA;
20 end while
21 return best solution from population in MA;
```

In general, an approach of this type has several problems that are mentioned in the following:

- the difficulty of defining a prediction model that adjusts with an acceptable confidence to the behavior of the human user; in fact this corresponds with the difficulty of finding a measure of the adequate distance that captures the subjective preferences of the human user;
- the inherent noise that often exists in the human response (due for instance to the fatigue of user, to the evolution of their subjective perception, or to an adjustment of its response to the characteristics of the solutions in the current generation);
- the difficulty to evaluate the level of fatigue (i.e., represented as the function `fatigue` in Algorithm 2) associated to the human user as the optimization process evolves. This is not an easy task and a primitive solution consists of imposing a maximum number of user interventions that has been agreed previously.

In any case, the flexibility of the proactive approach makes it helpful in cases in which the user wants to obtain an added value, but makes it also useful in complex optimization problems with perfectly well defined evaluation functions; in these cases the inherent skills of perception and information processing of the human user can help to both lead the search towards suboptimal regions of the search space and avoid the stagnation (or even premature convergence) of the algorithm in specific parts of this space. In the following section we present a study case in the context of a Gene Ordering Problem (GOP).

5.1 Study case: Gene Ordering Problem

This section describes the application of a number of MAs (including interactive and proactive proposals) on the *gene ordering problem* (Cotta et al. 2003), an NP-hard problem with strong implications in biomedicine.

5.1.1 Rationale

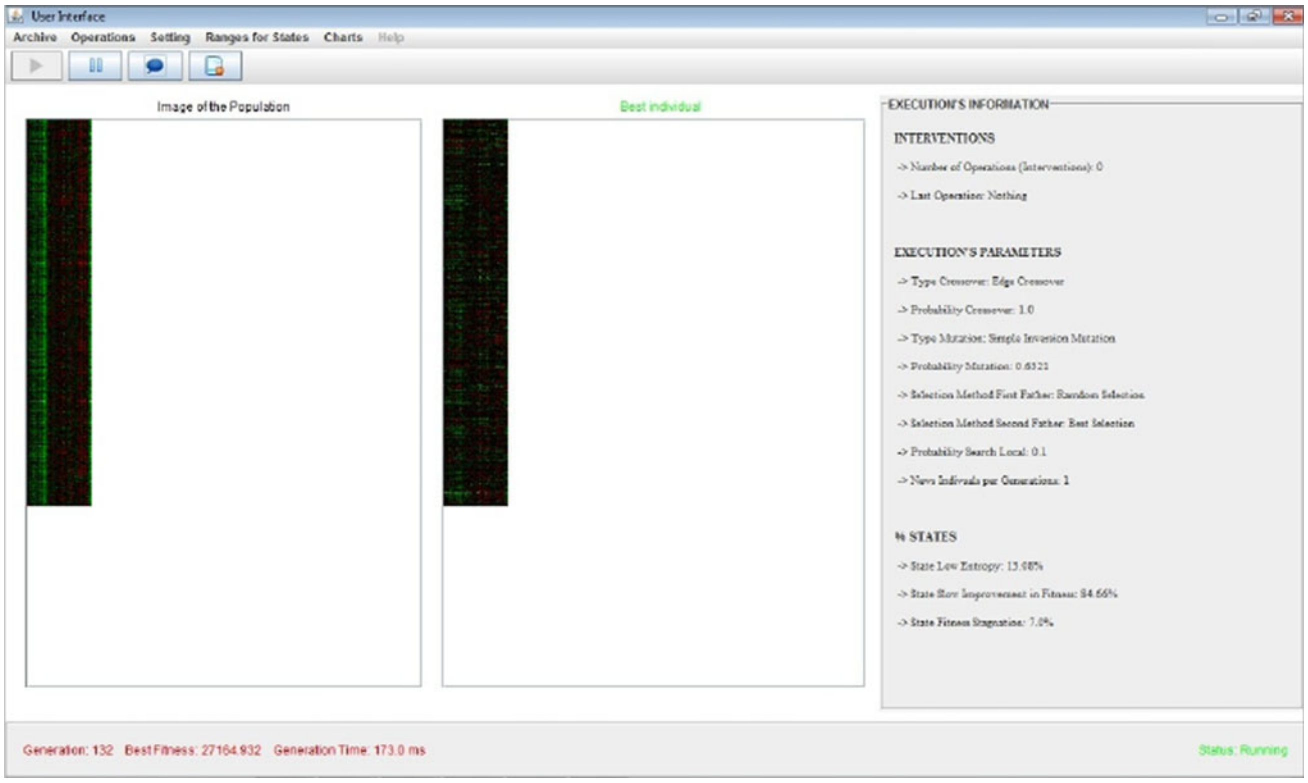
Thanks to microarray technology (De Risi et al. 1997), biologists can monitor the activity of hundreds up to tens of thousands of genes, with usually tens of measurements per gene. As a result, a data deluge takes place very much demanding reduction techniques [e.g., genes are believed to be influenced on average by about eight to ten other genes (Arnone and Davidson 1997)]. To this end genes with related expression patterns are grouped together since such genes are likely to regulate each other, or be co-regulated. Clustering techniques (Ben-Dor and Yakhini 1999; Eisen et al. 1998; Fasulo 1999; Hartuv et al. 1999) can be used, but this does not exhaust the possibilities. The Gene Ordering Problem (GOP) address this issue aiming to obtaining a high-quality re-arrangement of gene-expression data, such that related (from the point of view of their expression level) genes be placed in nearby locations within a gene sequence.

The result of a microarray experiment can be expressed as a matrix $G = \{g_{ij}\}$, $i = 1...n$, $j = 1...m$, where n is the number of genes, and m is the number of experiments per gene. The GOP amounts to finding an optimal order of genes such that genes with similar expression patterns are close in this order. For this purpose a notion of distance among genes is required. For simplicity we can consider the Euclidean distance: $D[g_i, g_j] = \left[\sum_{k=1}^{m}\left(g_{ik} - g_{jk}\right)^2\right]^{1/2}$. Once this distance matrix is found, fitness is computed by calculating the total distance between adjacent genes, similarly to what is done in the Traveling Salesman Problem. Thus, if $\pi = \langle\pi_1, \pi_2, \ldots, \pi_n\rangle$ is the gene ordering, the total distance between adjacent genes can be described as the $\sum_{i=1}^{n-1} D[\pi_i, \pi_{i+1}]$ (other fitness functions are possible, see Cotta et al. 2003).

5.1.2 Experiments

The user interface for the GOP is similar to that shown for the TSP—see Fig. 6. Basically, a graphical depiction of the best individual and an average composition of the population state is provided, along with controls for modifying along the run every parameter or element of the algorithm (e.g., check Fig. 7). The LS is conducted by selecting a portion of the best individual and checking whether exchanges of adjacent positions leads to a fitness improvement. In addition, the user also has the possibility of performing alterations such as for example inverting a portion of the image, and freezing/unfreezing a part of the solution (which will be then left unaltered by evolutionary operators).

In order to mitigate user fatigue the MA proactively suggests actions based on previous interventions of the user. These are recorded along with some indications of the state of the run at the point in which these actions were taken (the user can decide to leave some actions out of the record if she considers these actions were not valuable). The state of the run can be described in many different ways depending on different factors and the level of detail desired. In this case and for the sake of simplicity we have characterized the search state just in terms of three descriptors: diversity (population entropy), stagnation (number of iterations without improvement) and convergence speed (slope of the best-fitness curve in the last iterations), that range in an interval [0 %,100 %] (i.e., 0 % indicates the lowest value and 100 % the highest one; for stagnation we imposed a maximum number of iterations without improvement). When the user decides to intervene,



EXECUTION'S INFORMATION

INTERVENTIONS

-> Number of Operations (Interventions): 0

-> Last Operation: Nothing

EXECUTION'S PARAMETERS

-> Type Crossover: Edge Crossover

-> Probability Crossover: 1.0

-> Type Mutation: Simple Inversion Mutation

-> Probability Mutation: 0.6321

-> Selection Method First Father: Random Selection

-> Selection Method Second Father: Best Selection

-> Probability Search Local: 0.1

-> News Indivuals per Generations: 1

% STATES

-> State Low Entropy: 15.98%

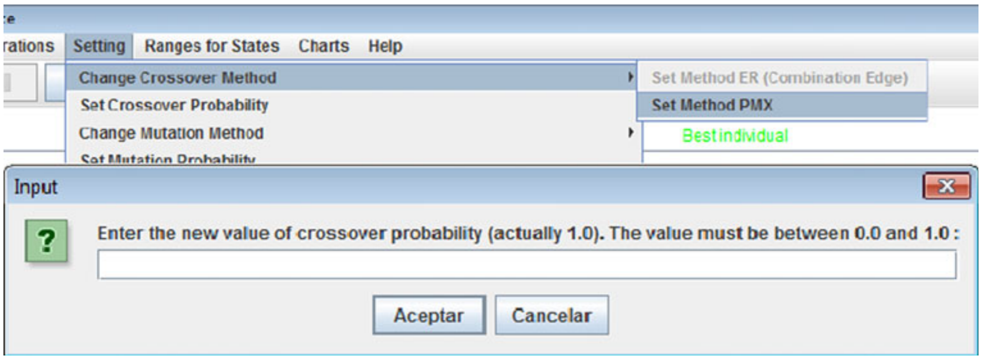
-> State Slow Improvement in Fitness: 84.66%

-> State Fitness Stagnation: 7.0%

Generation: 132 Best Fitness: 27164.932 Generation Time: 173.0 ms Status: Running

Fig. 6 Interface for the GOP

Fig. 7 Example of user intervention for the GOP



Enter the new value of crossover probability (actually 1.0). The value must be between 0.0 and 1.0 :

the current state is compared against recorded states and the action that best fit is chosen and suggested to the user who has the last word on whether it should be applied or not (an automatic always-accept mode can be used as well). More specifically, and as already mentioned, in the proactive model learning comes from previous experiences so that each time the algorithm is executed it will take into account its past executions. During each execution, the *conflictive states* (i.e., phases of the algorithm that clearly do not hold desirable properties –e.g., high diversity of the population, acceptable ratio of solution improvement, and non-premature convergence– according to the descriptors mentioned above) are registered beside the actions that were specifically applied with the aim of changing the state nature (to non-conflictive), as well as statistical information about how many times this action was taken under the same state and the amount of times that its application was successful (i.e., it allowed to reverse the situation to a

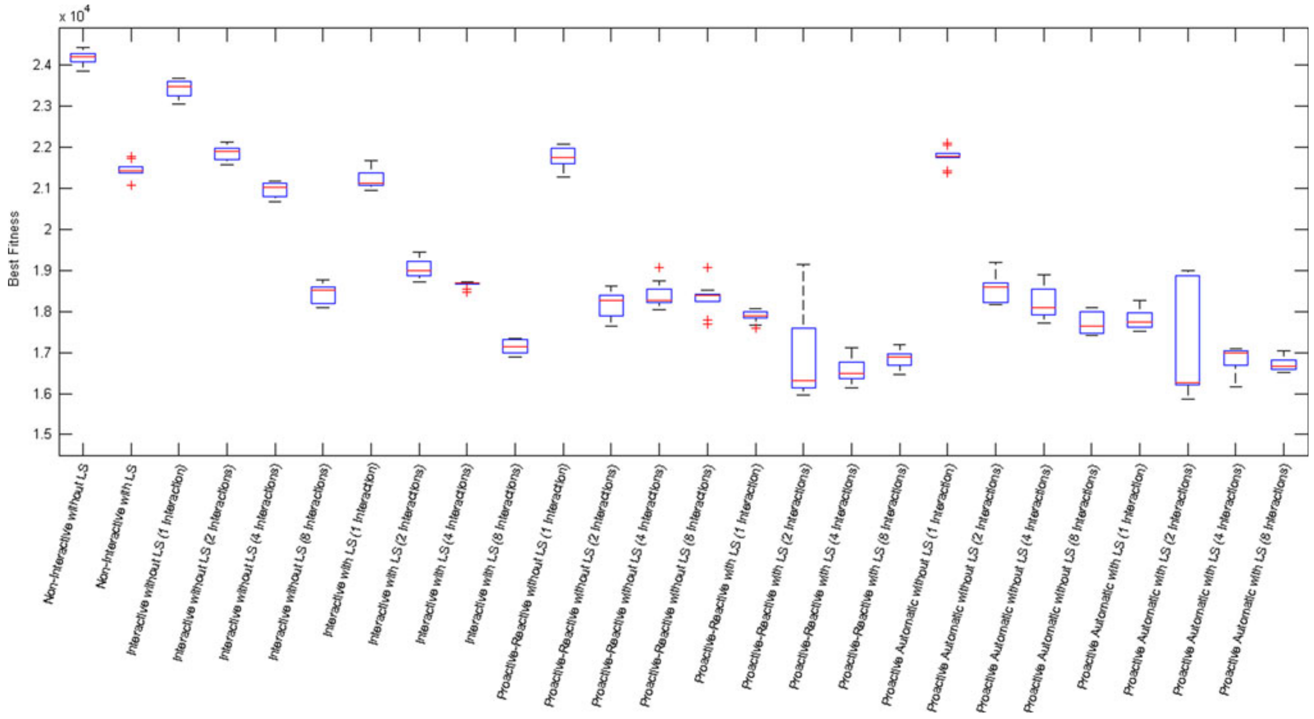


Fig. 8 Results on a dataset comprising selectively expressed genes in diffuse large B-cell lymphoma (380 genes, 19 experiments per gene) (Alizadeh et al. 2001). Algorithms in *X* axis, from left to right: non-interactive without LS, non-interactive with LS, interactive without LS (1 interaction), interactive without LS (2 interactions), interactive without LS (4 interactions), interactive without LS (8 interactions), interactive with LS (1 interaction), interactive with LS (2 interactions), interactive with LS (4 interactions), interactive with LS (8 interactions), proactive-reactive without LS (1 interaction), proactive-reactive without LS (2 interactions), proactive-reactive without LS (4 interactions), proactive-reactive without LS (8 interactions), proactive-reactive with LS (1 interaction), proactive-reactive with LS (2 interactions), proactive-reactive with LS (4 interactions), proactive-reactive with LS (8 interactions), proactive-automatic without LS (1 interaction), proactive-automatic without LS (2 interactions), proactive-automatic without LS (4 interactions), proactive-automatic without LS (8 interactions), proactive-automatic with LS (1 interaction), proactive-automatic with LS (2 interactions), proactive-automatic with LS (4 interactions), and proactive-automatic with LS (8 interactions)

non-conflictive state). Moreover, each state is also associated to a quantitative value of *unrest* that represents the amount (i.e., percentage) of conflict (measured as the average of the values of the descriptors associated to the state) that the state exhibits. This value is also recorded as part of the past experience. All this information is used by the algorithm to propose new actions in the future; basically, once a new conflictive state is detected the algorithm will try to apply, with a probability directly related with its percentage of unrest, the best action that, under equal circumstances, was shown to be successful in the past, and register this information in memory for further feedback. Those operations that in the past were discarded by the user in similar circumstances have also less probability to be elected for application.

Two problem instances have been considered for the experiments: a dataset comprising selectively expressed genes in diffuse large B-cell lymphoma (380 genes, 19 experiments per gene) (Alizadeh et al. 2001), and a dataset describing Kaposi's sarcoma-associated herpes virus gene expression (106 genes, 21 experiments per gene) (Jenner et al. 2001).

The MA has been used in four different settings: no interaction, interactive (with different number of user interventions; a single user–different from those used in the TSP experiments and more specialized in this problem–has been considered), proactive-reactive (the MA suggests an action to be done) and automatic proactive (the MA simulates user interventions in previous runs). Ten runs of each algorithm are done, using a steady-state MA, *maxevals* = 10,000, *popsize* = 40, edge recombination, and mutation by block inversion.

First of all, Fig. 8 shows the results on a 380-gene instance. Firstly we can observe that the application of LS is useful as all the MAs outperform its corresponding non-memetic versions, independently of the incorporation of interactivity. Also, regarding user interactions, as it can be seen, we observe again that all the interactive versions performs better than their corresponding non-interactive counterparts; this noticeable result can be a clear indication of the utility of incorporating human knowledge during the execution of the algorithms; note also that there is a general trend of improved results when the number of user interventions increases. Moreover the proactive versions are generally better than the purely interactive ones, this indicating that the ability of the MA for suggesting actions is valuable for the user.

This result is further confirmed by the results on a 106 instance (Fig. 9), in which the MA with no user input is compared to the proactive automatic MA which uses the

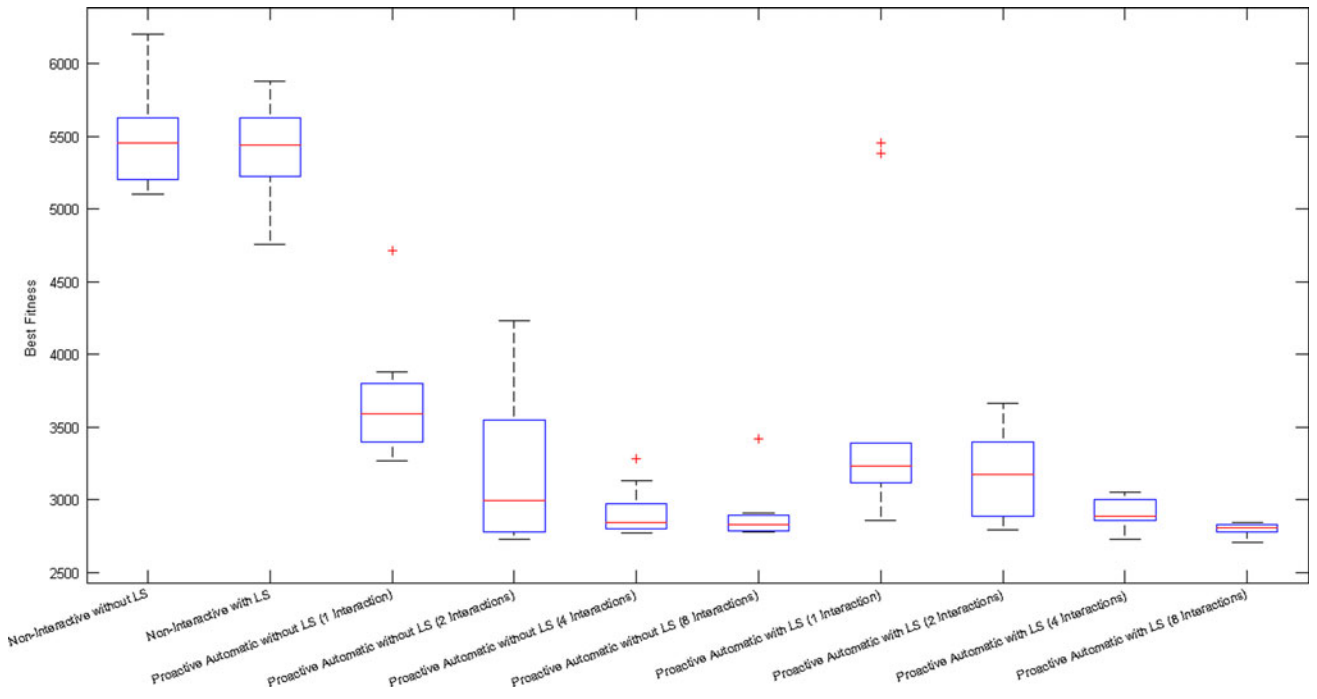


Fig. 9 Results on a dataset describing Kaposi's sarcoma-associated herpes virus gene expression (106 genes, 21 experiments per gene) (Jenner et al. 2001). Algorithms in X axis, from left to right: non-interactive without LS, non-interactive with LS, proactive automatic without LS (1 interaction), proactive automatic without LS (2 interactions), proactive automatic without LS (4 interactions), proactive automatic without LS (8 interactions), proactive automatic with LS (1 interaction), proactive automatic with LS (2 interactions), proactive automatic with LS (4 interactions), and proactive automatic with LS (8 interactions)

experience gathered in the previous instance. The same trends are observed, indicating that the MA can successfully apply the lessons learned on the previous instance to a unseen instance of the same problem (obviously, this need not be the case on other problems in which particular features of the instance varied wildly).

Particularly, as regard the proactive algorithms and considering the two problem instances, in general we did not find significant differences (statistically speaking, and at the standard level of 5 % level using a Wilcoxon rank-sum test) between the reactive versions and their corresponding automatic equivalents. This is an important result that encourages the employment of predictive models as real alternative to the human expert that might be replaced by the user model (i.e., as shown in Fig. 5) without decreasing the performance of the algorithm; moreover, this result suggests that, once the human user is disconnected from the search process, the execution might be totally automated. Two considerations though should be done here: (1) even for the proactive automatic algorithms presented here, the intervention of a human expert is required (either in the initial stages of the algorithm execution or in previous executions of this—or similar interactive algorithms, perhaps without proactivity—over the same problem instance) to construct a historical record of past experiences so that the predictive model is constructed from this. (2) The cost of obtaining a predictive model should be taken into account before implementing a proactive automatic algorithm as the attainment of a predictive model (as presented in this paper) demands a number of previous interventions of the human; in any case it seems clear that proactivity is a worthwhile mechanism when the problem demands a high level of expertise and fatigue is an important factor to decrease.

6 Conclusions

User-centric EC is an thriving research topic. Paving the way for further extensions, we have conducted in this work a study on the deployment of interactive capabilities in a MA, with application to two complex NP-hard problems. The results have been encouraging, since it has been shown that even some forms of limited interaction are capable of improving the results of a baseline autonomous algorithm. While the computational scenario is not a tough one, these results indicate that these techniques are capable of taking advantage from good-quality human feedback, not merely as a carrier of subjective information but as a source of problem-aware perturbations that can drive/focus the algorithm towards specific regions of the search space. At any rate, much remains to be done. As mentioned before, IEC is merely the tip of the iceberg; full-fledged user-centric optimization also implies proactivity in the search heuristic, anticipating the needs of the user, or trying to follow her preferences in order to provide hints in the direction she is headed to. Our results using some simple models of proactive behavior have also yielded encouraging results. We are currently working on some related user-modeling areas in the context of videogames, from which some general lessons will be hopefully learned.

Acknowledgments This work is partially supported by Spanish MICINN under projects NEMESIS (TIN2008-05941) and ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project P10-TIC-6083 (DNEMESIS).

References

Abu-Mostafa Y (1993) Hints and the VC dimension. Neural Comput 5:278–288

Arnone A, Davidson B (1997) The hardwiring of development: organization and function of genomic regulatory systems. Development 124:1851–1864

Alizadeh A et al (2001) Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403: 503–511

Babbar M, Minsker B (2006) A collaborative interactive genetic algorithm framework for mixed-initiative interaction with human and simulated experts: a case study in long-term groundwater monitoring design. In: World environmental and water resources congress

Bonissone PP, Subbu R, Eklund NHW, Kiehl TR (2006) Evolutionary algorithms + domain knowledge = real-world evolutionary computation. IEEE Trans Evol Comput 10(3):256–280

Breukelaar R, Emmerich M, Bck T (2006) On interactive evolution strategies. In: Rothlauf F, Branke J, Cagnoni S, Costa E, Cotta C, Drechsler R, Lutton E, Machado P, Moore J, Romero J, Smith G, Squillero G, Takagi H (eds) Applications of evolutionary computing. Lecture notes in computer science, vol 3907, Springer, Berlin, pp 530–541

Beck JC, Wilson N (2005) Proactive algorithms for scheduling with probabilistic durations. In: Proceedings of the 19th international joint conference on Artificial intelligence. IJCAI'05. Morgan Kaufmann, San Francisco, pp 1201–1206

Beck JC, Wilson N (2007) Proactive algorithms for job shop scheduling with probabilistic durations. J Artif Intell Res 28(1):183–232

Ben-Dor A, Yakhini Z (1999) Clustering gene expression patterns. In: Proceedings of the ACM RECOMB'99, Lyon, France. ACM Press, New York, pp 33–42

Cotta C, Fernández Leiva AJ (2011) Bio-inspired combinatorial optimization: notes on reactive and proactive interaction. In: Cabestany J, Rojas I, Caparrós GJ (eds) Advances in computational intelligence—11th international work-conference on artificial neural networks, Part II (IWANN 2011). Lecture notes in computer science, vol 6692. Springer, Málaga, pp 348–355

Cotta C, Troya JM (2003) Embedding branch and bound within evolutionary algorithms. Appl Intell 18(2):137–153

Cotta C, Mendes A, Garcia V, França P, Moscato P (2003) Applying memetic algorithms to the analysis of microarray data. In: Raidl G et al (eds) Applications of evolutionary computing. Lecture notes in computer science, vol 2611. Springer, Berlin, pp 22–32

Culberson J (1998) On the futility of blind search: an algorithmic view of "no free lunch". Evol Comput 6(2):109–128

Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York

Dawkins R (1976) The selfish gene. Clarendon Press, Oxford

Dawkins R (1986) The BlindWatchmaker, 1986. Longman, Essex

Deb K, Chaudhuri S (2007) I-mode: an interactive multi-objective optimization and decision-making using evolutionary methods. KanGal report 2007003, Kanpur Genetic Algorithms Laboratory

Deb K, Kumar A (2007) Interactive evolutionary multi-objective optimization and decision-making using reference direction method. KanGal report 2007001, Kanpur Genetic Algorithms Laboratory

De Risi J, Lyer V, Brown P (1997) Exploring the metabolic and genetic control of gene expression on a genomic scale. Science 278:680–686

Dias J, Captivo M, Clímaco J (2008) A memetic algorithm for multi-objective dynamic location problems. J Global Optim 42:221–253

Dozier G (2001) Evolving robot behavior via interactive evolutionary computation: from real-world to simulation. In: 16th ACM symposium on applied computing (SAC2001), Las Vegas, NV. ACM Press, New York, pp 340–344

Eiben AE, Smith JE (2003) Introduction to evolutionary computation. Springer, Berlin

Eisen M, Spellman P, Brown P, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci USA 95:14863–14868

Espinar J, Cotta C, Fernández-Leiva AJ (2012) User-centric optimization with evolutionary and memetic systems. In: Lirkov I, Margenov S, Wasniewski J (eds) 8th international conference on large-scale scientific computing (LSSC 2011). Lecture Notes in Computer Science, Sozopol, Bulgaria, vol 7116. Springer, Berlin, pp 214–221

Fasulo D (1999) An analysis of recent work on clustering algorithms. Technical Report UW-CSEO1-03-02, University of Washington

Gallardo J, Cotta C, Fernández A (2007) On the hybridization of memetic algorithms with branch-and-bound techniques. IEEE Trans Syst Man Cybern Part B 37(1):77–83

Gong D, Yao X, Yuan J (2009) Interactive genetic algorithms with individual fitness not assigned by human. J Univ Comput Sci 15(13):2446–2462

Hart WE, Belew RK (1991) Optimizing an arbitrary function is hard for the genetic algorithm. In: Belew RK, Booker LB (eds) Proceedings of the fourth international conference on genetic algorithms, San Mateo CA. Morgan Kaufmann, San Francisco, pp 190–195

Hart W, Krasnogor N, Smith J (2005) Recent advances in memetic algorithms. Studies in fuzziness and soft computing, vol 166. Springer, Berlin

Hartuv E, Schmitt A, Lange J, Meier-Ewert S, Lehrach H, Shamir R (1999) An algorithm for clustering cDNAs for gene expression analysis. In: Proceedings of the ACM RECOMB'99, Lyon, France. ACM Press, New York, pp 188–197

Houck C, Joines J, Kay M, Wilson J (1997) Empirical investigation of the benefits of partial lamarckianism. Evol Comput 5(1):31–60

Inoue T, Furuhashi T, Fujii M, Maeda H, Takaba M (1999) Development of nurse scheduling support system using interactive EA. IEEE Int Conf Syst Man Cybern 5:533–537

Jaszkiewicz A (2004) Interactive multiple objective optimization with the pareto memetic algorithm. In: Gottlieb J et al (eds) 4th EU/ME workshop: design and evaluation of advanced hybrid meta-heuristics, Nottingham, UK

Jenner R, Alba M, Boshoff C, Kellam P (2001) Kaposi's sarcoma-associated herpesvirus latent and lytic gene expression as revealed by DNA arrays. J Virol 75:891–902

Khanna R, Liu H, Chen HH (2008) Proactive power optimization of sensor networks. In: IEEE international conference on communications (ICC), Beijing, China, IEEE, pp 2119–2123

Klau G, Lesh N, Marks J, Mitzenmacher M (2010) Human-guided search. J Heuristics 16:289–310

Kosorukoff A (2001) Human-based genetic algorithm. In: 2001 IEEE international conference on systems, man, and cybernetics. IEEE Press, Tucson, pp 3464–3469

Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Comput 9(5):474–488

Kubota N, Nojima Y, Sulistijono I, Kojima F (2003) Interactive trajectory generation using evolutionary programming for a partner robot. In: 12th IEEE international workshop on robot and human interactive communication (ROMAN 2003), Millbrae, California, USA, pp 335–340

Lim S, Cho SB (2005) Language generation for conversational agent by evolution of plan trees with genetic programming. In: Torra V, Narukawa Y, Miyamoto S (eds) Modeling decisions for artificial intelligence. Lecture notes in computer science, vol 3558. Springer, Berlin, pp 305–315

Lim S, Kim KM, Hong JH, Cho SB (2004) Interactive genetic programming for the sentence generation of dialogue-based travel planning system. In: 7th Asia-Pacific conference on complex systems, Cairns, Australia. Asia-Pacific Workshops on Genetic Programming, pp 6–10

Lozano JA, Larrañaga P, Inza I, Bengoetxea E (2006) Towards a new evolutionary computation: advances on estimation of distribution algorithms. Studies in fuzziness and soft computing, vol 192. Springer, Berlin

Mamoun MH (2010) A new proactive routing algorithm for manet. Int J Acad Res 2(2):199–204

Moscato P (1999) Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization, McGraw-Hill, Maidenhead, pp 219–234

Moscato P, Cotta C (2003) A gentle introduction to memetic algorithms. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 105–144

Moscato P, Cotta C (2007) Memetic algorithms. In: Gonzalez TF (eds) Handbook of approximation algorithms and metaheuristics, Chapter 27. Chapman & Hall, London

Moscato P, Cotta C (2010) A modern introduction to memetic algorithms. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. International series in operations research and management science. 2nd edn, vol 146. Springer, Berlin, pp 141–183

Moscato P, Mendes A, Cotta C (2004) Memetic algorithms. In: Onwubolu G, Babu B (eds) New optimization techniques in engineering. Springer, Berlin, pp 53–85

Mühlenbein H, Paaß G (1996) From recombination of genes to the estimation of distributions I. Binary parameters. In: PPSN IV: Proceedings of the 4th international conference on parallel problem solving from nature, London, UK. Springer, Berlin, pp 178–187

Neri F, Cotta C (2012) Memetic algorithms and memetic computing optimization: a literature review. Swarm Evol Comput 2:1–14

Neri F, Cotta C, Moscato P (2012) Handbook of memetic algorithms. Studies in computational intelligence, vol 379. Springer, Berlin

Nguyen QH, Ong YS, Krasnogor N (2007) A study on the design issues of memetic algorithm. In: Srinivasan D, Wang L (eds) 2007 IEEE congress on evolutionary computation, Singapore, IEEE Computational Intelligence Society. IEEE Press, New York, pp 2390–2397

Ong YS, Keane A (2004) Meta-lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110

Ohsaki M, Takagi H, Ohya K (1998) An input method using discrete fitness values for interactive ga. J Intell Fuzzy Syst 6(1):131–145

Ong YS, Lim MH, Zhu N, Wong K (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern Part B 36(1):141–152

Parmee IC (2007) Human-centric evolutionary systems in design and decision-making. In: Rennard JP (eds) Handbook of research on nature-inspired computing for economics and management. IGI Global, pp 395–411

Parmee I, Abraham J (2004) User-centric evolutionary design. In: Marjanovic D (eds) 8th international design conference DESIGN 2004. Decision making workshop, pp 1441–1446

Parmee IC, Abraham JAR, Machwe A (2008) User-centric evolutionary computing: melding human and machine capability to satisfy multiple criteria. In: Knowles J, Corne D, Deb K, Chair DR (eds) Multiobjective problem solving from nature. Natural computing series. Springer, Berlin, pp 263–283

Puchinger J, Raidl GR (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In: Mira J, Álvarez JR (eds) Artificial intelligence and knowledge engineering applications: a bioinspired approach. First international work-conference on the interplay between natural and artificial computation, (IWINAC 2005), Part II. LNCS, vol 3562. Springer, Las Palmas, pp 41–53

Quiroz JC, Banerjee A, Louis SJ (2008) Igap: interactive genetic algorithm peer to peer. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. GECCO '08. ACM, New York, pp 1719–1720

Quiroz J, Louis S, Banerjee A, Dascalu S (2009) Towards creative design using collaborative interactive genetic algorithms. In: IEEE congress on evolutionary computation (CEC 2009), Singapore, IEEE, pp 1849–1856

Sáez Y, Viñuela PI, Segovia J, Castro JCH (2005) Reference chromosome to overcome user fatigue in IEC. New Gener Comput 23(2)

Smith JE (2008) Self-adaptation in evolutionary algorithms for combinatorial optimisation. In: Cotta C, Sevaux M, Sörensen K (eds) Adaptive and multilevel metaheuristics. Studies in computational intelligence, vol 136. Springer, Berlin, pp 31–57

Sudholt D (2009) The impact of parametrization in memetic evolutionary algorithms. Theor Comput Sci 410(26):2511–2528

Takagi H (2000) Active user intervention in an ec search. In: 5th Joint conference information sciences (JCIS2000), Atlantic City, NJ, pp 995–998

Takagi H (2001) Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. Proc IEEE 9:1275–1296

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82