Chapter 1
Memetic Algorithms and Complete Techniques

Carlos Cotta, Antonio J. Fernández Leiva, and José E. Gallardo

1.1 Introduction

As mentioned in previous chapters in this volume, metaheuristics (and specifically MAs) have a part of their raison d'etre in practically solving problems whose resolution would be otherwise infeasible by means of other non-heuristic approaches. Such alternative non-heuristic approaches are complete methods that –unlike heuristics– do guarantee that the deviation from optimality of the solution they will provide is somehow bounded (and as a particular case, that the optimal solution will be found). These methods are eventually limited by the curse of dimensionality, yet they may still constitute a very interesting resource either from the application point of view, or from the lessons that can be learnt from them. Indeed, in some sense these approaches could be considered complementary to metaheuristics rather that mere "rivals". Even more so in the case of MAs, whose philosophy has been since its inception much more flexible and integrative rather than dogmatic or exclusive.

This said, despite the eclosion of metaheuristics as powerful optimization techniques during the 80s and 90s, inter-breeding between the fields of provably problem-solving and heuristic problem-solving was relatively limited until the last decade (some seminal works dating back from the mid 90s – e.g., [10]). The last years have however witnessed a remarkable increase in the number of works trying to combine ideas from these two areas. Certainly, MAs have also played an important role in this cross-fertilization of search paradigms. Along this chapter we will review some of the lines of research that have emerged in this regard. To this end, we will begin by briefly revisiting complete techniques to highlight their strengths and weaknesses, and what they have to offer to metaheuristics. Subsequently, we will outline some

Carlos Cotta · Antonio J. Fernández Leiva · José E. Gallardo
Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga,
ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain
e-mail: {ccottap, afdez, pepeg}@lcc.uma.es

of the efforts that have been made in the literature to classify hybrid approaches. Although these classifications are usually general and intended to cover more than just complete-heuristic combinations, they will provide a framework within which actual combinations of MAs with exact techniques (or from the broad interpretation of memetic algorithms, MAs incorporating exact techniques) can be studied. This will be done in Sections 1.4 and 1.5.

1.2 Background

Complete techniques are those whose results can be proved to be at bounded distance from the optimum. From a very general point of view, these techniques can be further subdivided into techniques that guarantee finding the optimal solutions, i.e., exact techniques, and techniques that only provide a fixed or adjustable bound (that is, a bound that can be reduced by spending more computational effort), i.e., approximation techniques. Curiously, and this is something that may be worth some further analysis from a sociological and/or philosophical point of view, the community of researchers working on approximation theory has been traditionally more skeptical with respect to the value of metaheuristic optimization. Conversely, it is also true that the usefulness of approximation algorithms has not been always appreciated by the metaheuristic community, in part due to the inherent limitations of the former in many practical contexts – see for example [13] for a glimpse of the computational complexity of PTAS (a polynomial time approximation scheme, probably one of the jewels of the crown in approximation theory) for several common problems.

Focusing thus on exact techniques, such as for example branch and bound [26], dynamic programming [2], branch and cut [31], etc. these are characterized by the fact that they guarantee finding optimal solutions at the cost of a non-polynomial growth of computation time (and often memory consumption too). Their limitations are those emanating from the theory of computational complexity, such as the conspicuous P vs NP question. It must be noted however that such classical (unidimensional) hardness characterizations are not necessarily correlated with practical performance. A much more interesting characterization can be obtained from the field of parameterized complexity [12], in which hardness is approached from a multidimensional perspective, factoring out some *parameter*(s) from the input and trying to isolate the problem's difficulty in them. If this can be done –formally, if the complexity of the problem can be shown to be polynomially related to the input size (once the parameter is factored out), and the degree of the polynomial is unrelated to the value of the parameter– the problem is said to be fixed-parameter tractable (FPT). FPT problems can be solved for small values of the parameter using the arsenal developed by the parameterized-complexity community – e.g., [33]. Hard problems can be nevertheless detected from a parameterized perspective, and for such problems metaheuristics are fully in order.

There are many ways in which the hybridization of metaheuristics in general (and MAs in particular) with exact techniques can be fruitful: exact techniques can

for example reduce their resource consumption if they obtain valuable input from metaheuristics (e.g., improved bounds); on the other hand, metaheuristics routinely use search mechanisms –recombination, mutation, etc.– in which exact techniques can play an important role to intensify the search. Furthermore, hybridization of MAs and exact techniques can be defined at several nested levels thus providing multiple ways of boosting each other's performance. In the following we will survey some successful hybridization models reported in the literature along these lines just depicted. Previously, we will overview several approaches to classify these hybrid models.

1.3 Classification of Hybridization Approaches

Several taxonomical attempts have been proposed to classify hybrid optimization algorithms. For example, Talbi [41] proposed a mixed hierarchical-flat classification scheme. The hierarchical component captured the structure of the hybrid, whereas the flat component specified the features of the algorithms involved in the hybrid. More precisely, the hierarchical portion of the taxonomy firstly distinguished between low-level (a given function of a metaheuristic is replaced by another metaheuristic) and high-level (combined algorithms are self-contained) hybridization. Secondly, it was distinguished between relay hybridization (a set of metaheuristics is applied in a pipeline fashion) and teamwork hybridization (cooperative optimization models). Cotta [6] proposed another related taxonomy with the dichotomy *strong* vs. *weak* as its root. This distinction referred to whether problem-knowledge was placed in the core of the algorithm, affecting its internal components (e.g., representation and/or genotype-phenotype mapping, operators, etc.), or in the combination of different search algorithms that retained their identity. This terminology is consistent with the classification of problem-solving strategies in artificial intelligence as *strong* and *weak* methods [29].

A much more interesting classification for the purposes of this chapter is that proposed by Puchinger and Raidl [35]. This classification is specifically intended for exact-metaheuristic combinations, and establishes two main categories for such hybrid algorithms:

- *Collaborative combinations*, where an exact algorithm and a metaheuristic method exchange some information, but none of them are part of the other, and
- *Integrative combinations*, where one technique is a subordinate of the other, i.e., there is a *master* algorithm that uses the other one.

These two categories can be further refined depending on the particular of the combination as shown inf Figure 1.1. Thus, a collaborative combination can be sequential or parallel/intertwined, depending on how the control flow passes from one algorithm to the other. Similarly, an integrative combination can be subdivided in

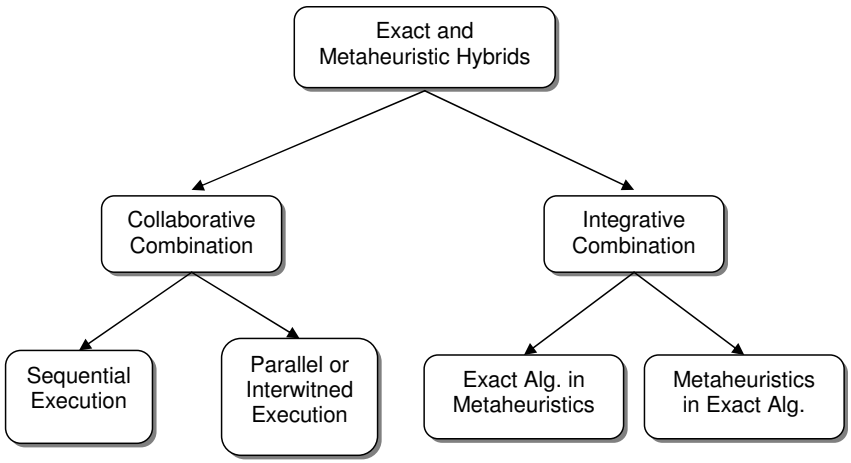

Fig. 1.1: Puchinger and Raidl's classification of exact-heuristic hybrid algorithms.

models in which an exact technique plays the role of master (i.e., the metaheuristic is embedded in an exact technique), and models in which the opposite is true.

As mentioned before, this latter classification fits nicely context of this chapter, so we will consider it in order to survey existing hybrid approaches combining MAs and exact techniques.

1.4 Integrative Combinations

One basic form of integrative collaboration consists of endowing a memetic algorithm with an exact technique (ET) so that this ET is a subordinate of the MA. The most common implementation consists of combining an EA with a procedure to perform a complete local search (which can consider the whole neighborhood and in this sense can be viewed as an exact technique). This 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 an ET inside a MA is to provide specific knowledge that can help to a better optimization process. For instance, Algorithm 1 shows a general picture of where an ET can be incorporated inside an MA.

As it can be seen, during the initialization of the population some complete method may be used to generate high quality initial solutions. Of course, this complete method may only consider a subset of the search space, a relaxed version of the problem, or may perform just a truncated search, since otherwise the problem would just be solved at that stage (not to mention the computational cost). An example of relaxed initialization using complete techniques can be found in [7], where a backtracking algorithm is used to create feasible initial solutions for a protein structure

Algorithm 1: Pseudocode of a basic MA based on a integrative collaboration with an exact technique ET

```
1  for i ∈ {1,...,POPULATION SIZE} do
2      pop[i] ←RANDOM-SOLUTION();
3      if Rand[0,1] < p_ET then // ET is applied with probability p_ET
4          EXACT-TECHNIQUE (pop[i]); // Usually ET = Local Improvement
5      end if
6  end for
7  i ← 0;
8  while i < MaxEvals do
9      RANK-POPULATION (pop); // sort population according to fitness
10     parent_1 ←SELECT (pop);
11     if Rand[0,1] < p_X then // recombination is done
12         parent_2 ← SELECT (pop);
13         child ← RECOMBINE (parent_1, parent_2); // RECOMBINE might be an
               Exact Technique
14     else
15         child ← parent_1;
16     end if
17     child ← MUTATE (child, p_M); // p_M is the mutation probability per
           gene
18     if Rand[0,1] < p'_ET then // ET is applied
19         EXACT-TECHNIQUE (child); // Usually Local Improvement
               applied here
20     end if
21     pop[μ] ← child; // replace worst
22 end while
23 return best solution in pop;
```

prediction problem (thus relaxing optimality to mere feasibility). Another related approach will be discussed in next subsection in the context of collaborative models, and considers a variant of a B&B algorithm –namely beam search– to initialize the population of a MA with the aim of improving its performance. In addition to this an exhaustive LS could be applied to improve the individuals generated initially with the aim of providing a first population of better quality. This procedure is intimately related to the idea of local branching by Fischetti and Lodi [14], and to Congram's Dynasearch [4, 5].

Another proposal that can be devised from the general schema shown above is the use of an ET as a recombination operator. Recombination or mutation operators can be intelligently designed so that specific problem knowledge is used in order to improve the offspring. For instance, Cotta *et al.* [10] used a problem-specific B&B approach for the Travelling Salesman Problem based on 1-trees and the Lagrangean relaxation [42], to build a hybrid recombination operator. More precisely, the B&B was used in order to build the best possible tour within the (Hamiltonian) subgraph defined by the union of edges in the parents. This recombination procedure was costly, but provided better results than blind edge recombination. This model was later extended to a more general operator termed dynastically optimal recombination

(DOR) [9]. The term refers to the *dynastic potential*, which in the framework of Forma Analysis [37] denotes the set of children attainable from a certain set of parents. DOR thus consists of finding the best children in this dynastic potential, i.e., that with the best combination of parental features. This is done by "intelligently" exploring this set, using an adequate complete algorithm, check e.g. [8].

Related to the previous approach, [18] presented a memetic algorithm, embedded with tabu search, for weighted constraint satisfaction problems (see next section for a more detailed discussion of this kind of problems) in which bucket elimination (BE) [11] is used as a mechanism for recombining solutions, providing the best possible child from the parental set. BE is an exact technique related to dynamic programming which based on variable elimination and is commonly used for solving constraint satisfaction problems. This algorithm, with another collaborative proposals, was applied to the resolution of the maximum density still life problem, a hard constraint optimization problem based on Conway's game of life.

Additionally, problem knowledge can be incorporated in the genotype to phenotype mapping present in many MAs, like when repairing an infeasible solution. This technique is used for instance in the MA designed by Chu and Beasley [3] for the multidimensional 0-1 knapsack problem. The use of complete techniques, again relaxed or truncated, can be here considered as well, check, e.g., [7] for an example of using backtracking to repair infeasible solutions.

Another place where an exact method can be particularly useful when used inside an MA is in the optimization of problems where different representations are considered. In these cases, an exact technique can be specifically useful in the codification-decodification phase. For instance, Puchinger and Raidl [36] represented another attempt to incorporate exact methods in metaheuristics. This work considered different heuristics algorithms for a real world glass cutting problem and a combined GA and B&B approach was proposed. The GA used an order-based representation that was decoded with a greedy heuristic. Incorporating B&B in the decoding for occasionally (with a certain probability) locally optimizing subpatterns turned out to increase the solution quality in a few cases.

Note also that applying always the ET in each generation of the MA (or initially on each individual in the initial population) is not always the best option (as shown in [40] for the application of LS on each generated new individual). For instance, if one considers LS as the technique to embed inside a MA, partial Lamarckianism [23], namely applying local search only to a fraction of individuals, can result in better performance. These individuals to which local search will be applied can be selected in many different ways [32]. Thus, LS can be applied to improve the individual with certain probability p_{LS}; in case of application, the improvement uses up a number of *LSevals* evaluations (or in the case of specific local search such as HC, until it stagnates, whatever comes first). It is easy to extrapolate these results from the use of LS to any ET embedded in a MA.

In general, the underlying idea of this kind of integration is to combine the intensifying capabilities of the embedded ET 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 ET by themselves. Regarding this issue, one can find in the literature a number of proposals that explore the intensification/diversification balance within the memetic algorithm. 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 of the solutions being recombined.

The other possibility for integrative combinations is to incorporate a metaheuristics into an exact algorithm. One example is the hybrid algorithm combining Genetic Algorithms and Integer Programming B&B approaches to solve MAX-SAT problems described in [15]. This hybrid algorithm gathered information during the run of a linear programming-based B&B algorithm, and used it to build the population of an EA population. The EA was eventually activated, and the best solution found was used to inject new nodes in the B&B search tree. The hybrid algorithm was run until the search tree was exhausted, and hence it is an exact approach. However, in some cases it expands more nodes than the B&B algorithm alone.

1.5 Collaborative Combinations

As mentioned in Section 1.3, the class of *collaborative combinations* includes hybrid algorithms which exchange information, but such that none of them is a subordinate of the other. Two subcases can be here considered in order to execute both algorithms:

- *Sequential execution*, in which one of the algorithms is completely executed before the other. Examples of this group are those in which one of the techniques can act as a kind of preprocessing for the other or those where the result of one algorithm can be used as data to initialize the other.
- *Parallel or intertwined execution*, where both techniques are executed simultaneously, either in parallel (i.e., running at the same time on different processors) or in an intertwined way by alternating between both algorithms.

As an example of sequential combinations we can cite the work of Klau et al. [25], in which a branch and cut algorithm is used analogously to the idea of dynastically optimal recombination mentioned in previous section, to combine the final population provided by a MA. This hybrid algorithm is applied to the prize-collecting Steiner tree problem We will focus here on hybrid collaborative techniques in the second group. MAs and B&B techniques can be integrated by way of a *direct collaboration*, so that both techniques work alone in parallel (i.e., both processes perform independently) at the same level. Under this scheme, both processes will share the incumbent solution to the problem being solved. Whenever one of the algorithms finds a better approximation, it can update the solution. Two straightforward ways of obtaining a benefit of this parallel execution are [10]:

- The B&B algorithm can use the lower bound provided by the MA to purge its problem queue. Problems whose upper bound are smaller than the one obtained by the MA cannot improve the incumbent solution and can be safety removed.
- The B&B algorithm can provide information about more promising regions of the search space into the MA population. The aim of this process is to to guide the MA search towards these promising regions of the search space.

Several implementations of these schemes are possible. For example, Cotta *et al.* [10] proposed a collaborative approach such as the one described above for the TSP. Puchinger and Raidl [34] consider the parallel combination of a MA and a branch and cut algorithm for the multidimensional knapsack problem. The MA provides improved bounds to the branch and cut algorithm, and the latter provides both new best-so-far solutions and the corresponding dual variable values, to be used for repairing and local search. More recently, Gallardo *et al.* [17] defined a hybrid algorithm that starts by running a MA (with a randomly initialized population) in isolation, so that a first approximation to the solution is obtained. This initial solution is later used by a B&B algorithm to purge its problem queue. As it can be seen, no information from the B&B algorithm was used in this first execution of the MA. In a subsequent phase, the B&B algorithm starts its execution. New solutions found by the B&B are incorporated into the MA population (by replacing the worst individual). Whenever a new solution is found, the B&B phase is paused and the MA is run to stabilization. In addition, pending nodes in the B&B queue are incorporated into the MA population periodically. The intention of this transfer is to direct the MA to these regions of the search space, that represent the subset of the search space still unexplored by the Branch and Bound. In this way, the MA is used for finding probably good solutions in those regions. Upon finding an improved lower bound (or upon stabilization of the MA if no improvement is found), B&B is resumed. This process is repeated until the search tree is exhausted, or a time limit is reached. One interesting property of this hybrid algorithm is that it acts as an anytime algorithm, providing both a quasi-optimal solution, and an indication of the maximum distance to the optimum. In [16, 17] this implementation schema is used to tackle large instances of the multidimensional knapsack problem. Experimental results showed that the hybrid approach can provide high quality results, better than those obtained by the MA and B&B on their own.

An alternative implementation of the previous model consist on using beam search (BS)[1] instead of B&B. This is an incomplete derivative of the later and acts thus as an heuristic method. In essence, BS extends every partial solution from a set $\mathscr{B}$ (called the *beam*) in at most k_{ext} possible ways, generating a new beam. When all solutions in $\mathscr{B}$ have been extended, the algorithm reduces the new beam by selecting the best up to k_{bw} (called the *beam width*) solutions and proceeds. A very interesting feature of this heuristic is that it extends in parallel a set of different partial solutions in several possible ways. For this reason, it can be used to provide periodically diverse promising partial solutions to a population based search method such as a MA. A general description of the resulting hybrid algorithm is given in Algorithm 2.

Algorithm 2: Beam Search + MA hybrid algorithm

```
1  for l_0 levels do run BS;
2  ;
3  repeat
4      select popsize nodes from problem queue;
5      initialize MA population with selected nodes;
6      run MA;
7      if MA solution better than BS solution then
8          let BS solution ← MA solution;
9      end if
10     for l levels do run BS ;
11     ;
12 until timeout or tree-exhausted;
13 return BS solution;
```

The beam search part of the algorithm can be iterated for each level of the search tree that corresponds to the problem at hand. The hybrid algorithm starts by executing this process for an initial number of levels (parameter l_0 of the algorithm). Subsequently, both parts of the hybrid algorithm are alternatively executed until a termination condition is reached. Similarly to the first implementation, for every execution of the MA, its population is initialized using the nodes in the BS queue. As the size of the BS queue is usually larger than the MA population size, a criteria, such as selecting the best nodes according to some measure of quality or selecting a subset that provides high diversity, has to be used in order to select a subset from the queue. Nodes in the BS queue represent partial solutions in which some genes are fixed but others are indeterminate, so they must first be converted to full solutions in a problem dependent way. This must be considered when instantiating the general template for different combinatorial problems. This kind of collaborative integration of Beam Search and MAs has been used to tackle different combinatorial optimization problems. In [20] the hybrid algorithm was experimentally evaluated on the multidimensional 0-1 knapsack problem and on the shortest common supersequence problem, a NP-hard classical problem from the realm of string analysis. For both problems, it was shown the benefits of using the hybrid approach when compared to the constituents algorithms. Additionally, an analysis of the dynamics and sensitivity on different parameters of the algorithm was carried out. In [21], the hybrid algorithm was applied to the inference of phylogenetic trees, an important problem in Systematic Biology, that aims to represent the evolutionary history for a collection of organisms. That work focused in the ultrametric model for phylogenetic inference. A robust setting for the different parameters of the algorithm was determined, and the hybrid algorithm was experimentally shown to also be synergetic for this problem.

A related hybridization model has been defined in [22] for weighted constraint satisfaction problems (WCSP) [39]. A WCSP is a constraint satisfaction problem (CSP) in which preferences among solutions can be expressed. Formally, a WCSP can be defined by a tuple $(\mathscr{X}, \mathscr{D}, \mathscr{F})$, where $\mathscr{D} = \{D_1, \cdots, D_n\}$ is a set of *finite do-*

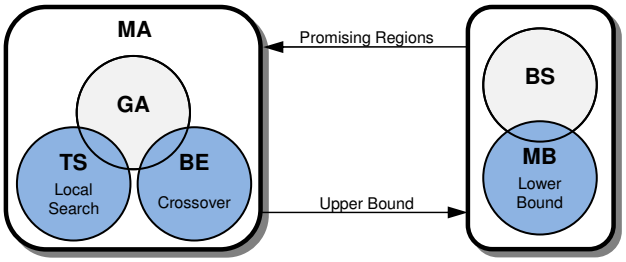

Fig. 1.2: Schematic description of the multilevel hybrid algorithm.

mains, $\mathscr{X} = \{x_1, \cdots, x_n\}$ is a set of variables taking values from their finite domains and $\mathscr{F}$ is a set of *cost functions* (also called *soft constraints* or *weighted constraints*) used to declare preferences among possible solutions. Each $f \in \mathscr{F}$ is defined over a subset of variables, $var(f) \subseteq \mathscr{X}$, called its *scope*. The objective function F –to be minimized– is defined as the sum of all functions in $\mathscr{F}$, i.e., $F = \sum_{f \in \mathscr{F}} f$. WCSP were tackled using a algorithmic model based on the hybridization of MAs with exact techniques at two levels: within the MA (as an embedded operator), and outside it (in a cooperative model). Figure 1.2 depicts the different components of the algorithm and their relationships. The first level of hybridization has already been described in Section 1.4, so we will describe here the second level, in which the MA cooperates with a beam search algorithm that further uses the technique of mini-buckets as a lower bound.

Algorithm 3: Hybrid algorithm for a WCSP

```
1  sol ← ∞;
2  ℬ ← { () };
3  for i ← 1 to n do
4  |   ℬ' ← {};
5  |   for s ∈ ℬ do
6  |   |   for a ∈ D_i do
7  |   |   |   ℬ' ← ℬ' ∪ {s · (x_i = a)} ;
8  |   |   end for
9  |   end for
10 |   ℬ ← select best k_bw nodes from ℬ';
11 |   if i ⩾ k_MA then
12 |   |   initialize MA population with best popsize nodes from ℬ';
13 |   |   run MA;
14 |   |   sol ← min (sol, MA solution);
15 |   end if
16 end for
17 return sol;
```

The proposed hybrid algorithm, that executes BS and the MA in an interleaved way, is depicted in Algorithm 3. Here, a (possibly partial) solution for a WCSP instance is represented by a vector of variables $s = (x_1, x_2, \ldots, x_i),\ i \leqslant n$, where $s \cdot (x_i = a)$ stands for the extension of partial solution s by assigning value a to its i-th variable. The hybrid algorithm proceeds by constructing a search tree, so that its leaves are complete solutions to the problem and internal nodes at level i represent solutions that are partially specified up to the i-th variable. The algorithm traverses this tree heuristically in a breadth first way using a BS algorithm that only maintains the best k_{bw} nodes at each level of the tree. During each iteration of BS (lines 5-16), a variable is assigned for every solution in the beam (line 8). The interleaved execution of the MA starts only when partial solutions in the beam have at least k_{MA} variables (line 12). For each iteration of BS, the best $popsize$ solutions in the beam are selected with the purpose of initializing the population of the MA (line 13). The solution provided after the execution of the MA is used to update the incumbent solution (sol), and this process is iterated until the search tree is exhausted.

The performance of this algorithm will depend on the quality of the heuristic function used to estimate partial solutions (line 11). In order to compute tight, yet computationally inexpensive, lower bounds for the remanning part of the solutions, the technique of mini-buckets (MB) can be used. As described by Kask and Dechter[24], the intermediate functions created by applying the MB scheme can be used as a general mechanism to compute heuristic functions that estimate the best cost of yet unassigned variables in partial solutions. This can be achieved by running MB as a preprocessing stage. The set of augmented buckets computed during this process can be used as estimations of the best cost extension to partial solutions (check [24] for details).

In [19, 22], such a multilevel algorithm was used to tackle the Maximum Density Still Life Problem, a hard constrained problem defined in the context of John Conway's game of life. The resulting algorithm was able to find optimal solutions for currently solved instances of the problem in considerable less time that state-of-the art approaches. Additionally, it was able to find new best known solutions for very large instances whose exact solutions are yet unknown.

1.6 Conclusions

Throughout this chapter we have surveyed existing work on MAs that incorporate at some level a complete technique. Several notes have to be done here. Notice firstly from a 'terminological' point of view that many evolutionary techniques hybridized with complete techniques can be considered memetic regardless of whether a classical trajectory-based local search algorithm is also used or not. For example, in an evolutionary algorithm that used an exact technique for recombination, the latter could be regarded as a generalized local-search operator working on set of solutions rather than on single solutions, and using a neighborhood composed of all solu-

tions in the corresponding dynastic potential. This is also related to the so-called crossover hill-climbing idea defined in [27] for continuous optimization.

From a practical point of view, this kind hybrid approaches must carefully control the computational complexity of the problems submitted to complete search. This draws again a connection to parameterized complexity by noting that this complexity is typically related to some structural parameter of the problem (e.g., a higher similarity of the parents during exact recombination reduces the size of the dynastic potential, thus making its exploration more amenable in principle). Even though no efficient (in the FPT sense) algorithmic resolution were available for the problem at hand, the combinatorial explosion could be kept within acceptable levels by checking these parameters and resorting to other approaches (truncated exact search, fast heuristic search, or even a blind procedure) if the possibility of a prohibitive computational cost cannot be excluded prior to a certain invocation of the complete method.

The any-time nature of MAs has to be considered as well. Whether a hybrid approach including complete techniques is itself complete or not, it is very important that it provides better and better solutions for any increasing computational budget allowed. This is not always possible within the context of complete techniques, e.g., a B&B algorithm using a best-first policy may exhaust its allotted time and/or memory without producing a single feasible solution. On the other hand, the very same B&B algorithm using a LIFO policy may quickly provide a solution but take a long time to improve it. MAs are however ideal for anytime search, and this can be exploited in a synergistic combination. Note for example that a parallel collaborative model using a MA and an exact technique may end up providing both an upper and a lower bound for the optimal solution, and the higher the computational budget available, the tighter these bounds will be.

Acknowledgements This work is supported by Spanish MICINN under project NEMESIS (TIN2008-05941) and Junta de Andalucía under project TIC-6083.

References

1. Barr A, Feigenbaum E (1981) Handbook of Artificial Intelligence. Morgan Kaufmann, New York NY
2. Bellman R (1957) Dynamic Programming. Princeton University Press, Princeton NJ
3. Chu PC, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics 4:63–86
4. Congram R (2000) Polynomially searchable exponential neighbourhoods for sequencing problems in combinatorial optimisation. PhD thesis, University of Southampton, Faculty of Mathematical Studies, UK
5. Congram R, Potts C, van de Velde S (2002) An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. INFORMS Journal on Computing 14(1):52–67
6. Cotta C (1998) A study of hybridisation techniques and their application to the design of evolutionary algorithms. AI Communications 11(3-4):223–224
7. Cotta C (2003) Protein structure prediction using evolutionary algorithms hybridized with backtracking. In: Mira J, Álvarez J (eds) Artificial Neural Nets Problem Solving Methods, Springer-Verlag, Berlin Heidelberg, Lecture Notes in Computer Science, vol 2687, pp 321–328
8. Cotta C, Troya J (2000) Using a hybrid evolutionary-A* approach for learning reactive behaviors. In: Cagnoni S, et al (eds) Real-World Applications of Evolutionary Computation, Springer-Verlag, Edinburgh, Lecture Notes in Computer Science, vol 1803, pp 347–356
9. Cotta C, Troya J (2003) Embedding branch and bound within evolutionary algorithms. Applied Intelligence 18(2):137–153
10. Cotta C, Aldana J, Nebro A, Troya J (1995) Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP. In: Pearson D, Steele N, Albrecht R (eds) Artificial Neural Nets and Genetic Algorithms 2, Springer-Verlag, Wien New York, pp 277–280
11. Dechter R (1999) Bucket elimination: A unifying framework for reasoning. Artificial Intelligence 113(1-2):41–85

12. Downey R, Fellows M (1995) Fixed parameter tractability and completeness I: Basic theory. SIAM Journal of Computing 24:873–921
13. Downey R, McCartin C (2005) Some new directions and questions in parameterized complexity. In: Calude C, Calude E, Dinneen M (eds) Developments in Language Theory, Springer-Verlag, Berlin Heidelberg, Lecture Notes in Computer Science, vol 3340, pp 12–26
14. Fischetti M, Lodi A (2003) Local branching. Mathematical Programmming 98:23–47
15. French A, Robinson A, Wilson J (2001) Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. Journal of Heuristics 7(6):551–564
16. Gallardo JE, Cotta C, Fernández AJ (2005) A hybrid model of evolutionary algorithms and branch-and-bound for combinatorial optimization problems. In: CEC 2005, IEEE Press, Edinburgh, UK, pp 2248–2254
17. Gallardo JE, Cotta C, Fernández AJ (2005) Solving the multidimensional knapsack problem using an evolutionary algorithm hybridized with branch and bound. In: [30], pp 21–30
18. Gallardo JE, Cotta C, Fernández AJ (2006) A memetic algorithm with bucket elimination for the still life problem. In: Gottlieb J, Raidl G (eds) Evolutionary Computation in Combinatorial Optimization, Springer-Verlag, Berlin Heidelberg, Lecture Notes in Computer Science, vol 3906, pp 73–85
19. Gallardo JE, Cotta C, Fernández AJ (2006) A multi-level memetic/exact hybrid algorithm for the still life problem. In: Runarsson TP, et al (eds) PPSN 2006, Springer-Verlag, Reykjavik, Iceland, Lecture Notes in Computer Science, vol 4193, pp 212–221
20. Gallardo JE, Cotta C, Fernández AJ (2007) On the hybridization of memetic algorithms with branch-and-bound techniques. IEEE Transactions on Systems, Man and Cybernetics, part B 37(1):77–83
21. Gallardo JE, Cotta C, Fernández AJ (2007) Reconstructing phylogenies with memetic algorithms and branch-and-bound. In: Bandyopadhyay S, Maulik U, Wang J (eds) Analysis of Biological Data: A Soft Computing Approach, World Scientific, pp 59–84
22. Gallardo JE, Cotta C, Fernández AJ (2009) Solving weighted constraint satisfaction problems with memetic/exact hybrid algorithms. Journal of Artificial Intelligence Research 35:533–555
23. Houck C, Joines J, Kay M, Wilson J (1997) Empirical investigation of the benefits of partial lamarckianism. Evolutionary Computation 5(1):31–60
24. Kask K, Detcher R (2001) A general scheme for automatic generation of search heuristics from specification dependencies. Artificial Intelligence 129:91–131
25. Klau G, Ljubić I, Moser A, Mutzel P, Neuner P, Pferschy U, Raidl G, Weiskircher R (2004) Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. In: Deb K, et al (eds) GECCO 2004, Springer-Verlag, Seattle WA, Lecture Notes in Computer Science, vol 3102, pp 1304–1315

26. Lawler E, Wood D (1966) Branch and bounds methods: A survey. Operations Research 4(4):669–719
27. Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evolutionary Computation 12(3):273–302
28. Merelo Guervós JJ, et al (eds) (2002) Parallel Problem Solving from Nature VII, Lecture Notes in Computer Science, vol 2439, Springer-Verlag, Granada, Spain
29. Michalewicz Z (1993) A hierarchy of evolution programs: An experimental study. Evolutionary Computation 1(1):51–76
30. Mira J, Álvarez J (eds) (2005) Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach, Lecture Notes in Computer Science, vol 3562, Springer-Verlag, Berlin Heidelberg
31. Nemhauser G, Wolsey L (1988) Integer and Combinatorial Optimization. John Wiley & Sons
32. Nguyen QH, Ong YS, Krasnogor N (2007) A study on the design issues of memetic algorithm. In: CEC 2007, IEEE Press, Singapore, pp 2390–2397
33. Niedermeier R, Rossmanith P (2000) A general method to speed up fixed-parameter-tractable algorithms. Information Processing Letters 73:125–129
34. Puchinger J, Raidl G (2005) Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem. In: Proceedings of the 2005 Metaheuristics International Conference, Vienna, Austria, pp 775–780
35. Puchinger J, Raidl GR (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In: [30], pp 41–53
36. Puchinger J, Raidl GR, Koller G (2004) Solving a real-world glass cutting problem. In: Gottlieb J, Raidl GR (eds) 4th European Conference on Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol 3004, pp 165–176
37. Radcliffe N (1994) The algebra of genetic algorithms. Annals of Mathematics and Artificial Intelligence 10:339–384
38. Ryan C, Keijzer M (eds) (2008) Genetic and Evolutionary Computation Conference – GECCO 2008, ACM Press, Atlanta GA
39. Schiex T, Fargier H, Verfaillie G (1995) Valued constraint satisfaction problems: hard and easy problems. In: 14th. International Join Conference on Artificial Intelligence, IJCAI-1995, Montreal, Canada, pp 631–637
40. Sudholt D (2009) The impact of parametrization in memetic evolutionary algorithms. Theoretical Computer Science 410(26):2511–2528
41. Talbi EG (2002) A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5):541–564
42. Volgenant A, Jonker R (1982) A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. European Journal of Operational Research 9:83–88

Index