Memetic Algorithms

P. Moscato

Newcastle Bioinformatics Initiative, University of Newcastle, Callaghan, NSW, 2308, Australia

E-mail: Pablo.Moscato@newcastle.edu.au

C. Cotta

ETSI Informática, University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain

E-mail: ccottap@lcc.uma.es

September 4, 2005

1 Introduction

The term *'Memetic Algorithms'* [74] (MAs) was introduced in the late 80s to denote a family of metaheuristics that have as central theme the hybridization of different algorithmic approaches for a given problem. Special emphasis was given to the use of a population-based approach in which a set of cooperating and competing agents were engaged in periods of individual improvement of the solutions while they sporadically interact. Another main theme was to introduce *problem and instance-dependent knowledge* as a way of speeding-up the search process. Initially, hybridizations included Evolutionary Algorithms –EAs [35, 41, 89, 97], Simulated Annealing and its variants [52] [79] and Tabu Search [75] [9]. Today, a number of hybridizations include other metaheuristics [42] as well as exact algorithms, in *complete anytime* memetic algorithms [76]. These methods not only prove optimality, they can deliver high-quality solutions early on in the process.

The adjective 'memetic' comes from the term 'meme', coined by R. Dawkins [30] to denote an analogous to the *gene* in the context of cultural evolution. It was first proposed as a mean of conveying the message that, although inspiring for many, biological evolution should not constrain the imagination to develop population-based methods. Other forms of evolution may be faster, with cultural evolution being one of those less-restrictive examples.

Due to the fact that MAs aimed at drawing the attention from two communities of researchers with different agendas, aiming at hybridizations of their methods, this metaheuristic had to suffer tough initial times. Today they are becoming increasingly popular due to their impressive success record and the high sophistication of the hybridizations proposed. As a consequence of its evolution, it is not unusual to find MAs disguised in the literature under different names such as 'hybrid EAs' or 'Lamarckian EAs'. Furthermore, many of the underlying ideas of MAs can be also found in other metaheuristics that evolved in relative isolation from MAs. Scatter search [38] is a good example of a metaheuristic sharing much of its functioning philosophy with MAs. In Ref. [80], the authors cite a paper by S. Kase in which a "game" between a set of hierarchical agents (players and referees) is proposed to hybridize heuristic approaches for a layout problem[50]. What makes this interesting is that this is an approach that does not rely on computers for optimization and helps the employees to become engaged in these issues. Anticipating further definitions, we can say that a MA is a search strategy in which a population of optimizing agents synergistically cooperate and compete [82]. A more detailed description of the algorithm, as well as an functional template will be given in Section 2.

As mentioned before, MAs is a hot topic nowadays, mainly due to their success in solving many hard optimization problems, attracting experienced researchers to work on the challenges of this field. A particular feature of MAs is greatly responsible for this: unlike traditional EAs, MAs are intrinsically concerned with exploiting *all available knowledge* about the problem under study. The advantages of this approach was notably neglected in EAs for a long time despite some contrary voices, most notably Davis' who also advocated for hybridization in his book [29]. The formulation of the so-called *No-Free-Lunch Theorem* (NFL) by Wolpert and Macready [104] made it definitely clear that a search algorithm strictly performs in accordance with the amount and quality of the problem knowledge they incorporate, thus backing up one of the *leiv motivs* of MAs.

MAs exploit problem-knowledge by incorporating pre-existing heuristics, preprocessing data reduction rules, approximation and fixed-parameter tractable algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc. Also, an important factor is the use of adequate representations of the problem being tackled. This results in highly efficient optimization tools. We provide a brief abstracted overview of MA applications in combinatorial optimization in Section 3. We will finish with a brief summary of the current research trends in MAs, with special mention to those we believe will play a major role in the near future.

2 Dissecting a Basic Memetic Algorithm

As mentioned in the previous section, MAs blend different search strategies in a combined algorithmic approach. Like EAs, MAs are population-based metaheuristics. This means that in MAs we maintain a *population* of solutions for the problem at hand. We are using the term "solutions" rather loosely here, as we can have either feasible or proto-solutions (structures that can be extended/modified to produce feasible solutions) or even unfeasible solutions (which can be "repaired" to restore feasibility). It is also assumed that both repairing or extension processes can be done quite fast, as to justify including them in the population. Each of these solutions will be termed *individual* as the EA jargon, mainly to simplify the discussion. In the context of MAs, the denomination *agent* representing a processing unit that can hold multiple solutions, and has problem-domain methods that help to improve them if required [74]. Each individual/agent represents a tentative solution/method for the problem under consideration. When the agents adapt their methods we call the resulting strategy an *adaptive memetic algorithm*. Adaptation may include a modification of the data as in [42].

Due to the agents interactions, solutions are subject to processes of competition and mutual cooperation. The general structure of MAs is shown in Figure 1.1. aiming to highlight similarities with other population-based metaheuristics such as EAs. Relevant differences are nevertheless evident when we inspect the innards of the high-level blocks depicted in Figure 1.1. First of all, notice the existence of an initialization block. Standard EAs would simply generate $\mu = |pop|$ random solutions. This can be also done in MAs, but more sophisticated mechanisms are typically used as they are more useful. For example, some constructive heuristic can be utilized to produce high-quality solutions [102] [61]. Another possibility refers to the use of a local improving method, as illustrated in Fig. 1.2.

There is another interesting element in the flow chart shown in Figure 1.1: the *Re-start Population* process. This component is sometimes present in some EAs, but it is essential in MAs. Consider that the population may reach a state in which the generation of a new improved solution might be very unlikely. This could be the case when all solutions in the population are very similar to each other. In this situation of population convergence, it is better to refresh the population, rather than keeping the population constrained to a small region of the search space, probably expending most of the time resampling the same solutions [22]. This is specifically important in MAs since the inclusion of several knowledge-augmented components contribute to accelerate the convergence of the population. Typical criteria for determining population convergence are measuring the diversity of solutions –via Shannon's entropy [28] for instance– and bayesian decision-making [44]. In either case, and whenever the population is considered to have converged, re-starting can be done in different ways. One of these is shown in Figure 1.3: top individuals of the population are kept (a certain fraction p of the population; this value should not be very high since otherwise the population would obviously converge again in a very short time afterwards), and the remaining solutions are created from scratch, as it is done in the initialization phase.

The main functional block in the MA template is the *generational step* process. This is actually the part of the algorithm in which evolution of solutions takes place. Its internal structure is depicted in Figure 1.4. As it can be seen, there are three main components in this generational step: selection, reproduction, and update. The first one and the third one are responsible for the competition aspects of individuals in the population. Using the information provided by a problem-dependent guiding function (termed *fitness* function in the EA terminology), the goodness of individuals in *pop* is evaluated, and a sample of individuals is selected according to this goodness measure to help create new solutions. Essentially, this selection can be done using fitness-proportionate methods (the probability of selecting an individual is proportional to its fitness), and non-proportionate methods (selection is done on the basis of qualitative comparisons among individuals). The latter are being increasingly used, since they avoid some problems of the former (assumption of maximization, need of transformation for dealing with minimization, scaling problems, etc.). Among these, we can cite rank-based methods (the top in the rank of an individual, the higher its chances for being selected), and tournament-based methods (individuals are selected on the basis of a direct competition within small sub-groups of individuals).

As to update, this component takes care of maintaining the population at a constant size, or more properly, at a manageable size, since variable-size populations are not rare [34]. This is done by substituting some pre-existing individuals in *pop* by some of the new ones from *newpop*, using some specific criterion. Two major strategies are possible: the *plus* strategy in which the best μ individuals from $pop \cup newpop$ are kept, and the *comma* strategy in which the best μ from *newpop* are kept. In the latter case, if $|pop| = |newpop|$ then the update is termed *generational*; if $|newpop|$ is small (say $|newpop| = 1$), then we have a *steady-state* replacement (the worst $|newpop|$ solutions from *pop* are substituted). Other intermediate *generational gaps* are possible by selecting higher values of $|newpop|$.

We finally arrive to the reproduction stage, where new individuals (or agents) are created using the information existing in the population. More precisely, several reproductive *operators* (i.e., transformation functions) are used in a pipelined fashion, as illustrated in Figure 1.4. Reproductive operators are algorithms that be classified into two classes: unary operators and $n-$ary ($n > 1$) operators. Beginning with the former, two further types of operators are typically used, namely *mutation* operators, and *individual-improvement* operators (in many cases based on some form of local search). The latter were already mentioned before, e.g., in the initialization phase. As indicated by their name, their purpose is to improve the quality of a ceratin solution. In general, this is implemented via an iterative process whereby small modifications are introduced in a solution, and kept if they result in an effective quality improvement. This process is repeated until it can be determined that no further improvement is possible, until the amount of quality improvement is considered good enough, or –most typically– until a maximum number of improving attempts are performed. Hence, the process need not stop at an optimum for the individual-improver, and therefore characterizations of MAs as *"EAs working in the space of local-optima [with respect to a certain fitness landscape]"* are clearly restricting even the methods that originated the denomination [74] [73] and should be avoided. As to mutation, it is intended to generate new solutions by partly modifying existing solutions. This modification can be random –as it is typically the case in EAs– or can be endowed with problem-dependent information so as to bias the search to probably-good regions of the search space.

Non-unary operators are usually termed recombination operators. These operators constitute a distinctive added-value possibility of population-based search, and encapsulate the mutual cooperation among several individuals (typically two of them, but a higher number is possible). They generate new individuals using the information contained in a number of selected solutions called *parents*. If it is the case that the resulting individuals (the *offspring*) are entirely composed of information taken from the parents, then the recombination is said to be *transmitting* [87]. This property can be difficult to achieve for certain problem domains (the *Traveling Salesman Problem* –TSP– is a typical example). In those situations, it is possible to consider other properties of interest such as *respect* or *assortment*. The former refers to the fact that the recombination operator generate descendants carrying all *features* (i.e., basic properties of solutions with relevance for the problem attacked) common to all parents; thus, this property can be seen as a part of the *exploitative* side of the search. On the other hand, *assortment* represents the exploratory side of recombination. A recombination operator is said to be *assorting* if, and only if, it can generate descendants carrying any combination of compatible features taken from the parents. In either case, similarly to mutation, performing the combination of information in a problem-oriented way (rather than blindly) is crucial for the performance of the algorithm, see, e.g., [26, 81].

This description of recombination has introduced a crucial concept, namely, *relevant features*. By relevant features we mean the information pieces that can be extracted from solutions, exerting a direct influence on the quality of these. Consider that a certain solution can contain a high number of *atomic* information units, but only some of them are directly linked with quality. For example, a permutation π can be interpreted as a collection of positional information units, i.e., position i has value π_i. It also can be interpreted as a collection of adjacency information units, i.e., values a and b occur in adjacent positions of the permutation. It turns out that if the permutation is taken as a solution to the TRAVELING SALESMAN PROBLEM, the latter are indeed the relevant features, while for the FLOWSHOP SCHEDULING PROBLEM positional information is much more important [24]. The definition of operators manipulating the relevant features is one of the key aspects in the design of MAs.

There have been several attempts for quantifying how good a certain set of information units is for representing solutions for a specific problems. We can cite a few of them:

- *Minimizing epistasis*: epistasis can be defined as the non-additive influence on the guiding function of combining several information units (see [27] for example). Clearly, the higher this non-additive influence, the lower the absolute relevance of individual information units. Since the algorithm will be processing such individual units (or small groups of them), the guiding function turns out to be low informative, and prone to misguide the search.

- *Minimizing fitness variance* [87]: This criterion is strongly related to the previous one. The fitness variance for a certain information unit is the variance of the values returned by the guiding function, measured across a representative subset of solutions carrying this information unit. By minimizing this fitness variance, the information provided by the guiding function is less *noisy*, with the subsequent advantages for the guidance of the algorithm.

- *Maximizing fitness correlation*: In this case a certain reproductive operator is assumed, and the correlation in the values of the guiding function for parents and offspring is measured. If the fitness correlation is high, good solutions are likely to produce good solutions, and thus the search will gradually shift toward the most promising regions of the search space. Again, there is a clear relationship with the previous approaches; for instance, if epistasis (or fitness variance) is low, then solutions carrying specific features will have similar values for the guiding function; since the reproductive operators will create new solutions by manipulating these features, the offspring is likely to have a similar guiding value as well.

Obviously, the description of these approaches may appear somewhat idealized, but the underlying philosophy is well illustrated. For further advice on the design of MAs, the reader is referred to [77, 78].

3 MAs and Combinatorial Optimization

MAs constitute a extremely powerful tool for tackling combinatorial optimization problems. Indeed, MAs are state-of-the-art approaches for many such problems. Traditional NP Optimization problems constitute one of the most typical battlefields of MAs, and a remarkable history of successes has been reported with respect to the application of MAs to such problems. Combinatorial optimization problems (both single-objective and multi-objective [45][47][54]) arising in scheduling, manufacturing, telecommunications, and bioinformatics among other fields have been also satisfactorily tackled with MAs. Some of these applications are summarized in Table 1.1.

This list of applications is by no means complete since its purpose is simply to document the wide applicability of the approach for combinatorial optimization. Indeed, MAs have been successfully applied to other domains. Other such application areas of MAs include machine learning, robotics, engineering, electronics, bioinformatics, oceanography, and many more. For further information about MA applications we suggest checking Refs. [77][78], or querying bibliographical databases or web browsers for the keywords *'memetic algorithms'* and *'hybrid genetic algorithm'*.

4 Conclusions and Future Directions

We believe that MAs have very favorable perspectives for their development and widespread application. We ground our belief in several reasons: firstly, MAs are showing a great record of efficient implementations, providing very good results in practical problems – cf. previous section. We also have reasons to believe that we are near some major leaps forward in our theoretical understanding of these techniques, including for example the computational complexity analysis of recombination procedures. On the other hand, the inherent asynchronous parallelism of MAs adapts very well to the increasing availability of distributed systems.

We also see as a healthy sign the systematic development of other particular optimization strategies. If a simpler –non-population-based– metaheuristic performs the same as a more complex method (GAs, MAs, Ant Colonies, etc.), Ockham's razor should prevail and we must either resort to the simpler method, or to the one that has less free parameters, or to the one that is easier to implement. Such a fact should defy us to adapt the complex methodology to beat a simpler heuristic, or to check if that is possible at all. An unhealthy sign of current research, however, are the attempts to encapsulate metaheuristics on stretched confinements. The evolutionary computing community had to endure a difficult time in the past, until the artificial boundaries among the different EA families were overcome. It would be unwise to repeat the same mistakes in the wider context of metaheuristics.

There are many open lines of research in MAs. One of them is multi-level evolution. It was anticipated in [76] that future MAs could simultaneously evolve solutions (in a short-time scale), as well as representations and methods (ia a longer-time scale). In this sense, Krasnogor has recently introduced techniques to adaptively change the neighborhood definition [57], and with colleagues is these adaptive memetic algorithms for the difficult problem of *protein structure prediction* [55]. Smith also presents a recent study on these issues in [99] and [94],

Multiparent recombination is another promising area in which further work has to be done. Recall that recombination is precisely one of the additional search possibilities contributed by population-based algorithms, and that its augmentation with problem knowledge results in notably enhanced optimization capabilities. It seems natural to generalize these ideas to multiple-solution recombination. Not only one can have a wider pool of information for building the offspring, but additional hints can be obtained with respect to, e.g., negative knowledge, that is, what pieces of information should be avoided in the offspring. This is definitely one of the most challenging issues for future development in MAs.

References

[1] H.S. Abdinnour. A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 106(2-3):489–99, 1998.

[2] C.C. Aggarwal, J.B. Orlin, and R.P. Tai. Optimized crossover for the independent set problem. *Operations Research*, 45(2):226–234, 1997.

[3] A. Alkan and E. Ozcan. Memetic algorithms for timetabling. In *2003 Congress on Evolutionary Computation*, pages 1796–1802, Canberra, Australia, 2003. IEEE Press.

[4] F. Altiparmak, B. Dengiz, and A.E. Smith. Optimal design of reliable computer networks: A comparison of metaheuristics. *Journal of Heuristics*, 9(6):471–487, Dec. 2003.

[5] M.J. Bayley, G. Jones, P. Willett, and M.P. Williamson. Genfold: A genetic algorithm for folding protein structures using NMR restraints. *Protein Science*, 7(2):491–499, 1998.

[6] A. Bazzoli and A. Tettamanzi. A memetic algorithm for protein structure prediction in a 3d-lattice hp model. In G.R. Raidl et al., editors, *EvoWorkshops*, volume 3005 of *Lecture Notes in Computer Science*, pages 1–10. Springer Verlag, 2004.

[7] J. Beasley and P.C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):393–404, 1996.

[8] J. Berger, M. Salois, and R. Begin. A hybrid genetic algorithm for the vehicle routing problem with time windows. In R.E. Mercer and E. Neufeld, editors, *Advances in Artificial Intelligence. 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 114–127, Berlin, 1998. Springer-Verlag.

[9] R. Berretta, C. Cotta, and P. Moscato. Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm: Results on the number partitioning problem. In M. Resende and J. Pinho de Sousa, editors, *Metaheuristics: Computer-Decision Making*, pages 65–90. Kluwer Academic Publishers, Boston MA, 2003.

[10] R. Berretta and P. Moscato. The number partitioning problem: An open challenge for evolutionary computation ? In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 261–278. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[11] R. Berretta and L.F. Rodrigues. A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics*, 87(1):67–81, Jan. 8 2004.

[12] M.J. Blesa, P. Moscato, and F. Xhafa. A memetic algorithm for the minimum weighted k-cardinality tree subgraph problem. In Jorge Pinho de Sousa, editor, *Proceedings of the 4th Metaheuristic International Conference (MIC'2001), Porto, Portugal, July 16-20, 2001*, pages 85–90, 2001.

[13] T.N. Bui and B.R. Moon. GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(3):193–204, 1998.

[14] L. Buriol, P.M. França, and P. Moscato. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506, Sep. 2004.

[15] L. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A memetic algorithm for OSPF routing. In *Sixth INFORMS Telecommunications Conference, March 10-13, 2002 Hilton Deerfield Beach, Boca Raton, Florida*, pages 187–188, 2002.

[16] E.K. Burke and A.J. Smith. A memetic algorithm for the maintenance scheduling problem. In *Proceedings of the ICONIP/ANZIIS/ANNES '97 Conference*, pages 469–472, Dunedin, New Zealand, 1997. Springer-Verlag.

[17] S. Cagnoni et al., editors. *Applications of Evolutionary Computing 2002*, volume 2279 of *Lecture Notes in Computer Science*. Springer Verlag, 2002.

[18] R. Cheng, M. Gen, and Y. Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms. II. Hybrid genetic search strategies. *Computers & Industrial Engineering*, 37(1-2):51–55, 1999.

[19] P.C. Chu and J. Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24:17–23, 1997.

[20] P.E. Coll, G.A. Durán, and P. Moscato. On worst-case and comparative analysis as design principles for efficient recombination operators: A graph coloring case study. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 279–294. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[21] D. Costa. An evolutionary tabu search algorithm and the NHL scheduling problem. *INFOR*, 33(3):161–178, 1995.

[22] C. Cotta. On resampling in nature-inspired heuristics. In V. Botti, editor, *Proceedings of the 7^{th} Conference of the Spanish Association for Artificial Intelligence*, pages 145–154, 1997. In Spanish.

[23] C. Cotta. Scatter search and memetic approaches to the error correcting code problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture Notes in Computer Science*, pages 51–60, Berlin, 5-7 April 2004. Springer Verlag.

[24] C. Cotta and J.M. Troya. Genetic forma recombination in permutation flowshop problems. *Evolutionary Computation*, 6(1):25–44, 1998.

[25] C. Cotta and J.M. Troya. A hybrid genetic algorithm for the 0-1 multiple knapsack problem. In G.D. Smith, N.C. Steele, and R.F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms 3*, pages 251–255, Wien New York, 1998. Springer-Verlag.

[26] C. Cotta and J.M. Troya. Embedding branch and bound within evolutionary algorithms. *Applied Intelligence*, 18(2):137–153, 2003.

[27] Y. Davidor. Epistasis variance: A viewpoint on GA-hardness. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 23–35. Morgan Kaufmann, 1991.

[28] Y. Davidor and O. Ben-Kiki. The interplay among the genetic algorithm operators: Information theory tools used in a holistic way. In R. Männer and B. Manderick, editors, *Parallel Problem Solving From Nature II*, pages 75–84, Amsterdam, 1992. Elsevier Science Publishers B.V.

[29] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold Computer Library, New York, 1991.

[30] R. Dawkins. *The Selfish Gene*. Clarendon Press, Oxford, 1976.

[31] P. de Causmaecker, G. van den Berghe, and E.K. Burke. Using tabu search as a local heuristic in a memetic algorithm for the nurse rostering problem. In *Proceedings of the Thirteenth Conference on Quantitative Methods for Decision Making*, pages abstract only, poster presentation, Brussels, Belgium, 1999.

[32] N. Dellaert and J. Jeunet. Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research*, 38(5):1083–1099, 2000.

[33] R. Dorne and J.K. Hao. A new genetic local search algorithm for graph coloring. In A.E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature V*, volume 1498 of *Lecture Notes in Computer Science*, pages 745–754, Berlin, 1998. Springer-Verlag.

[34] F. Fernández, L. Vanneschi, and M. Tomassini. The effect of plagues in genetic programming: A study of variable-size populations. In C. Ryan et al., editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *Lecture Notes in Computer Science*, pages 317–326, Essex, 14-16 April 2003. Springer-Verlag.

[35] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley & Sons, New York, 1966.

[36] P.M. França, J.N.D. Gupta, A.S. Mendes, P. Moscato, and K.J. Veltink. Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers& Industrial Engineering*, 48(3):491–506, May. 2005.

[37] P.M. França, A.S. Mendes, and P. Moscato. Memetic algorithms to minimize tardiness on a single machine with sequence-dependent setup times. In *5th International Conference of the Decision Sciences Institute*, pages 1708–1710, Atlanta, GA, USA, 1999. Decision Sciences Institute.

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

[39] A.B. Hadj-Alouane, J.C. Bean, and K.G. Murty. A hybrid genetic/optimization algorithm for a task allocation problem. *Journal of Scheduling*, 2(4), 1999.

[40] M. Hifi. A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. *Journal of the Operational Research Society*, 48(6):612–622, 1997.

[41] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[42] D. Holstein and P. Moscato. Memetic algorithms using guided local search: A case study. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 235–244. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[43] E. Hopper and B. Turton. A genetic algorithm for a 2d industrial packing problem. *Computers & Industrial Engineering*, 37(1-2):375–378, 1999.

[44] M. Hulin. An optimal stop criterion for genetic algorithms: A bayesian approach. In Th. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 135–143, San Mateo, CA, 1997. Morgan Kaufmann.

[45] H. Ishibuchi and K. Narukawa. Some issues on the of implementation of local search in evolutionary multi-objective optimization. *Lecture Notes in Computer Science*, 3102:1246–1258, 2004.

[46] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, Apr. 2003.

[47] A. Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Annals of Operations Research*, 131(1-4):135–158, Oct. 2004.

[48] W.R. Jih and Y.J. Hsu. Dynamic vehicle routing using hybrid genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 453–458, Washington D.C., 1999. IEEE.

[49] S. Kaige, T. Murata, and H. Ishibuchi. Performance evaluation of memetic EMO algorithms using dominance relation-based replacement rules on MOO test problems. In *2003 IEEE International Conference on Systems, Man and Cybernetics*, pages 14–19, Washington, USA, 2003. IEEE Press.

[50] S. Kase and N. Nishiyama. An Industrial Engineering Game Model for Factory Layout. *The Journal of Industrial Engineering*, XV(3):148–150, 1964.

[51] Sandor Kersting, Günther R. Raidl, and Ivana Ljubic. A memetic algorithm for vertex-biconnectivity augmentation. In Cagnoni et al. [17], pages 102–111.

[52] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[53] G.W. Klau, I. Ljubic, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, G. Raidl, and R. Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. *Lecture Notes in Computer Science*, 3102:1304–1315, 2004.

[54] J.D. Knowles and D.W. Corne. M-paes: a memetic algorithm for multiobjective optimization. In *The 2000 Congress on Evolutionary Computation, San Diego, USA*, pages 325–332, 2000.

[55] N. Krasnogor, B. Blackburne, J.D. Hirst, and E.K. Burke. Multimeme algorithms for protein structure prediction. In *7th International Conference on Parallel Problem Solving from Nature - PPSN VII, September 7-11, 2002, Granada, Spain*, 2002.

[56] N. Krasnogor and J. Smith. A memetic algorithm with self-adaptive local search: TSP as a case study. In D. Whitley et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 987–994, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[57] N. Krasnogor and J. Smith. Multimeme algorithms for the structure prediction and structure comparison of proteins. In A.M. Barry, editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops*, pages 42–44, New York, 8 July 2002. AAAI.

[58] R.M. Krzanowski and J. Raper. Hybrid genetic algorithm for transmitter location in wireless networks. *Computers, Environment and Urban Systems*, 23(5):359–382, 1999.

[59] D. Levine. A parallel genetic algorithm for the set partitioning problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 23–35. Kluwer Academic Publishers, Boston, MA, USA, 1996.

[60] A. Lim, B. Rodrigues, and Y. Zhu. Airport gate scheduling with time windows. *ARTIFICIAL INTELLIGENCE REVIEW*, 24(1):5–31, Sep. 2005.

[61] S.J. Louis, X. Yin, and Z.Y. Yuan. Multiple vehicle routing with time windows using genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1804–1808, Washington D.C., 1999. IEEE Press.

[62] R. Maheswaran, S.G. Ponnambalam, and C. Aravindan. A meta-heuristic approach to single machine scheduling problems. *International Journal of Advanced Manufacturing Technology*, 25(7-8):772–776, 2005.

[63] A. Mendes, P. França, P. Moscato, and V. Garcia. Population studies for the gate matrix layout problem. In F.J. Garijo, J.C. Riquelme, and M. Toro, editors, *IBERAMIA*, volume 2527 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2002.

[64] A. Mendes and A. Linhares. A multiple-population evolutionary approach to gate matrix layout. *International Journal of Systems Science*, 35(1):13–23, Jan. 15 2004.

[65] A.S. Mendes, P.M. França, and P. Moscato. Fitness landscapes for the total tardiness single machine scheduling problem. *Neural Network World, an International Journal on Neural and Mass-Parallel Computing and Information Systems*, pages 165–180, 2002.

[66] A.S. Mendes, F.M. Muller, P.M. França, and P. Moscato. Comparing meta-heuristic approaches for parallel machine scheduling problems with sequence-dependent setup times. In *Proceedings of the 15th International Conference on CAD/CAM Robotics & Factories of the Future, Aguas de Lindoia, Brasil*, volume 1, pages 1–6, Campinas, SP, Brazil, 1999. Technological Center for Informatics Foundation.

[67] P. Merz. A comparison of memetic recombination operators for the traveling salesman problem. In W. B. Langdon et al., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 472–479, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[68] P. Merz. Analysis of gene expression profiles: an application of memetic algorithms to the minimum sum-of-squares clustering problem. *BIOSYSTEMS*, 72(1-2):99–109, Nov. 2003.

[69] P. Merz and B. Freisleben. A Genetic Local Search Approach to the Quadratic Assignment Problem. In T. Bäck, editor, *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 465–472, San Francisco, CA, USA, 1997. Morgan Kaufmann.

[70] P. Merz and B. Freisleben. Memetic Algorithms and the Fitness Landscape of the Graph Bi-Partitioning Problem. In A.-E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 765–774, Berlin, Germany, 1998. Springer.

[71] P. Merz and B. Freisleben. A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem. In Pete Angeline, editor, *1999 Congress on Evolutionary Computation (CEC'99)*, pages 2063–2070, Piscataway, NJ, USA, 1999. IEEE Press.

[72] P. Merz and B. Freisleben. Greedy and local search heuristics for the unconstrained binary quadratic programming problem. *Journal of Heuristics*, 8(2):197–213, 2002.

[73] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[74] P. Moscato. On genetic crossover operators for relative order preservation. C3P Report 778, California Institute of Technology, Pasadena, CA 91125, 1989.

[75] P. Moscato. An Introduction to Population Approaches for Optimization and Hierarchical Objective Functions: The Role of Tabu Search. *Annals of Operations Research*, 41(1-4):85–121, 1993.

[76] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[77] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.

[78] P. Moscato, C. Cotta, and A. Mendes. Memetic algorithms. In G.C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 53–85. Springer-Verlag, Berlin Heidelberg, 2004.

[79] P. Moscato and M. G. Norman. A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In M. Valero et al., editors, *Parallel Computing and Transputer Applications*, pages 177–186, Amsterdam, 1992. IOS Press.

[80] P. Moscato and F. Tinetti. Blending heuristics with a population-based approach: A memetic algorithm for the traveling salesman problem. Report 92-12, Universidad Nacional de La Plata, C.C. 75, 1900 La Plata, Argentina, 1992.

[81] Y. Nagata and Sh. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In Th. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms, East Lansing, EUA*, pages 450–457, San Mateo, CA, 1997. Morgan Kaufmann.

[82] M.G. Norman and P. Moscato. A competitive and cooperative approach to complex combinatorial search. Technical Report Caltech Concurrent Computation Program, Report. 790, California Institute of Technology, Pasadena, California, USA, 1989.

[83] J. Pacheco and O. Valencia. Design of hybrids for the minimum sum-of-squares clustering problem. *Computational Statistics & Data Analysis*, 43(2):235–248, Jun. 28 2003.

[84] B. Paechter, A. Cumming, M.G. Norman, and H. Luchian. Extensions to a Memetic timetabling system. In E.K. Burke and P. Ross, editors, *The Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 251–265. Springer Verlag, 1996.

[85] C. Prins and S. Bouchenoua. A memetic algorithm solving the VRP, the CARP and general routing problems with nodes, edges and arcs. In David Corne et al., editors, *Advances in Nature-Inspired Computation: The PPSN VII Workshops*, pages 12–13, Reading, UK, 2002. PEDAL (Parallel, Emergent & Distributed Architectures Lab), University of Reading.

[86] A. Quintero and S. Pierre. Sequential and multi-population memetic algorithms for assigning cells to switches in mobile networks. *Computer Networks*, 43(3):247–261, Oct. 22 2003.

[87] N.J. Radcliffe and P.D. Surry. Fitness Variance of Formae and Performance Prediction. In L.D. Whitley and M.D. Vose, editors, *Proceedings of the 3^{rd} Workshop on Foundations of Genetic Algorithms*, pages 51–72, San Francisco, 1994. Morgan Kaufmann.

[88] E. Ramat, G. Venturini, C. Lente, and M. Slimane. Solving the multiple resource constrained project scheduling problem with a hybrid genetic algorithm. In Th. Bäck, editor, *Proceedings of the Seventh*

International Conference on Genetic Algorithms, pages 489–496, San Francisco CA, 1997. Morgan Kaufmann.

[89] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[90] C. Reeves. Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63:371–396, 1996.

[91] A.M. Rodrigues and J. Soeiro Ferreira. Solving the rural postman problem by memetic algorithms. In Jorge Pinho de Sousa, editor, *Proceedings of the 4th Metaheuristic International Conference (MIC'2001), Porto, Portugal, July 16-20, 2001*, pages 679–684, 2001.

[92] S. Runggeratigul. A memetic algorithm to communication network design taking into consideration an existing network. In Jorge Pinho de Sousa, editor, *Proceedings of the 4th Metaheuristic International Conference (MIC'2001), Porto, Portugal, July 16-20, 2001*, pages 91–96, 2001.

[93] A. Sakamoto, X.Z. Liu, and T. Shimamoto. A genetic approach for maximum independent set problems. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E80A(3):551–556, 1997.

[94] E.E. Santos and E. Santos Jr. Reducing the computational load of energy evaluations for protein folding. In *The 4th IEEE Symposium on Bioinformatics and Bioengineering, Taichung, Taiwan*, pages 79–86, 2004.

[95] V. Schnecke and O. Vornberger. Hybrid genetic algorithms for constrained placement problems. *IEEE Transactions on Evolutionary Computation*, 1(4):266–277, 1997.

[96] J. Schonberger, D.C. Mattfeld, and H. Kopfer. Memetic algorithm timetabling for non-commercial sport leagues. *European Journal of Operational Research*, 153(1):102–116, Feb. 16 2004.

[97] H.-P. Schwefel. *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik.* Diplomarbeit, Technische Universität Berlin, Hermann Föttinger–Institut für Strömungstechnik, März 1965.

[98] M. Sevaux, A. Jouglet, and C. Oğuz. Combining constraint programming and memetic algorithm for the hybrid flowshop scheduling problem. In *ORBEL 19 th annual conference of the SOGESCI-BVWB*, Louvain-la-Neuve, Belgium, 27-28 January 2005.

[99] J.E. Smith. Protein structure prediction with co-evolving memetic algorithms. In *The 2003 Congress on Evolutionary Computation, Canberra, Australia*, pages 2346–2353, 2003.

[100] K. Sörensen and M. Sevaux. MA|PM: Memetic algorithms with population management. *Computers and Operations Research*, 2004. In Press, available online 26 November 2004.

[101] N. Speer, C. Spieth, and A. Zell. A memetic co-clustering algorithm for gene expression profiles and biological annotation. In *2004 Congress on Evolutionary Computation*, pages 1631–1638, Portland, USA, 2004. IEEE Press.

[102] P.D. Surry and N.J. Radcliffe. Inoculation to initialise evolutionary search. In T.C. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, number 1143 in Lecture Notes in Computer Science, pages 269–285. Springer-Verlag, 1996.

[103] R. Torres-Velázquez and V. Estivill-Castro. A memetic algorithm guided by quicksort for the error-correcting graph isomorphism problem. In Cagnoni et al. [17], pages 173–182.

[104] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[105] W.-C. Yeh. A memetic algorithm for the min k-cut problem. *Control and Intelligent Systems*, 28(2):47–55, 2000.

[106] Y. Zhu, A. Lim, and B. Rodrigues. Aircraft and gate scheduling with time windows. In *The 15th IEEE International Conference on Tools with Artificial Intelligence, Sacramento, USA*, pages 189–193, 2003.

[107] P. Zou, Z. Zhou, G. Chen, and X. Yao. A novel memetic algorithm with random multi-local-search: a case study of tsp. In *The 2004 Congress on Evolutionary Computation, Portland, USA*, pages 2335–2340, 2004.

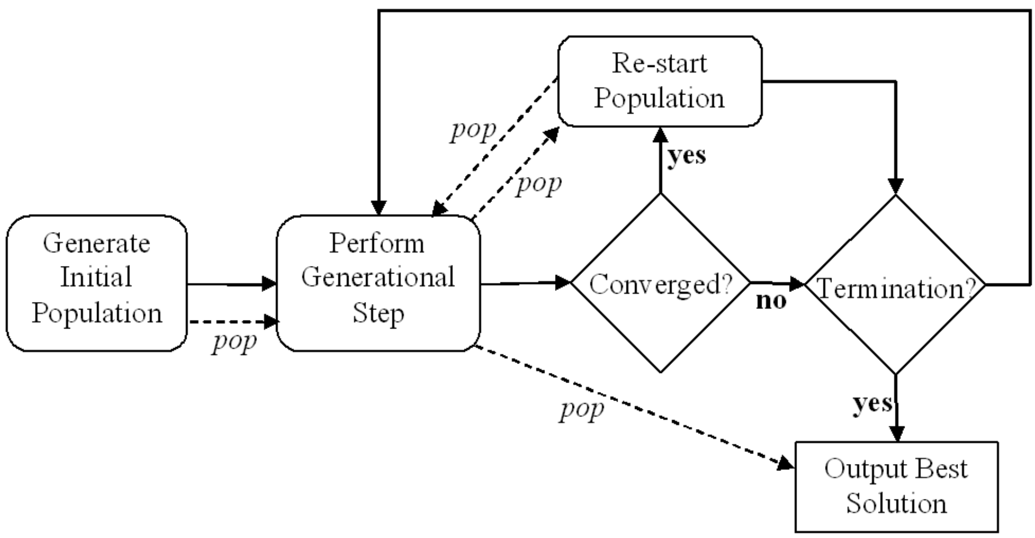

Figure 1.1: The general structure of MAs. Solid arrows indicate the control flow, whereas dashed arrows indicate the data flow.

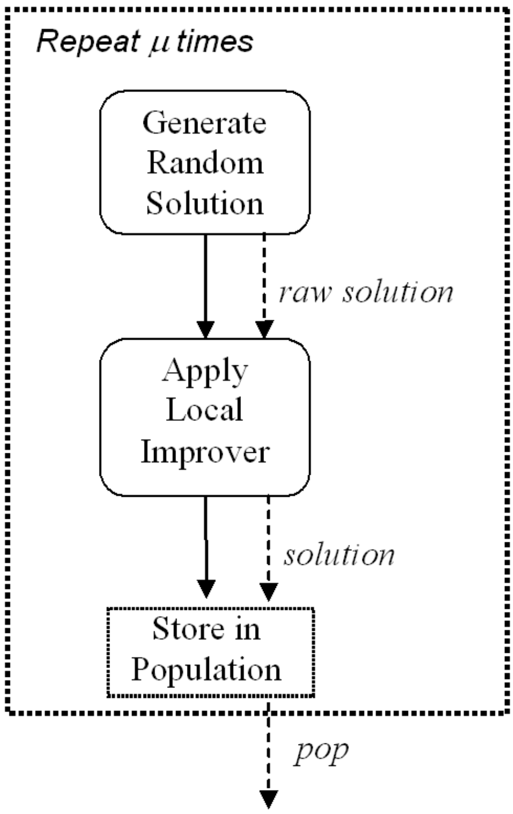

Figure 1.2: Generation of the initial population. A local improver can be used to enhance the quality of starting solutions.

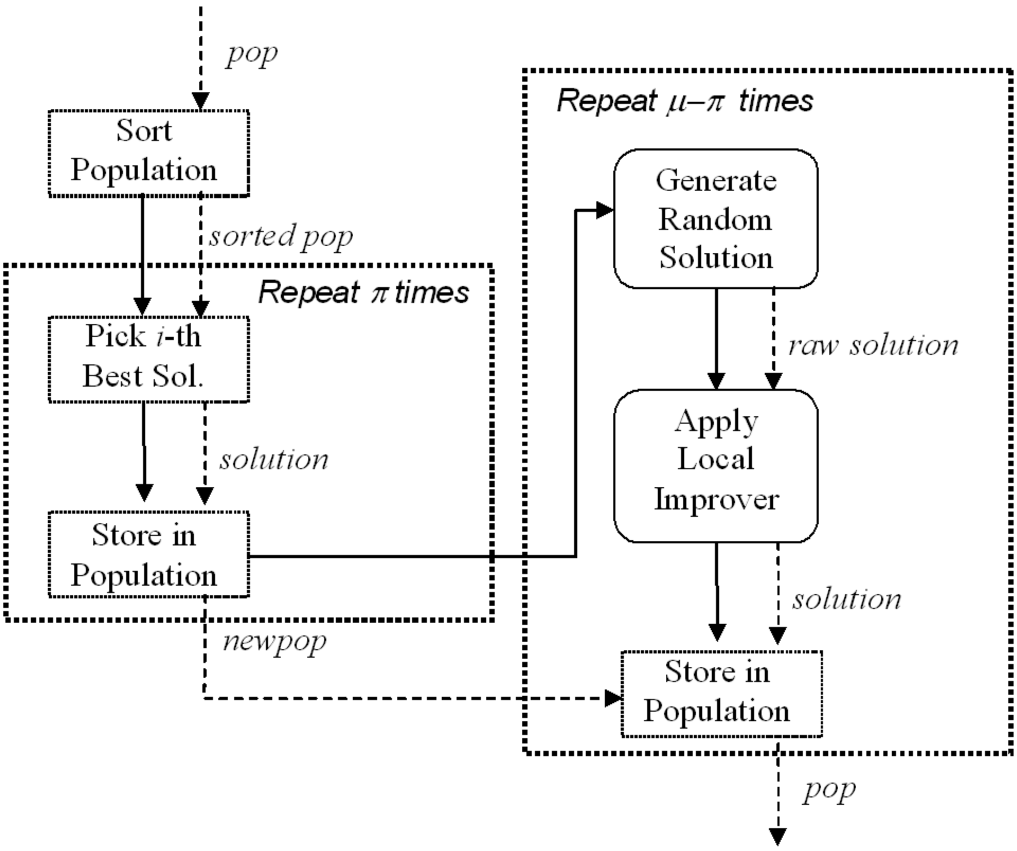

Figure 1.3: A possible re-starting procedure for the population. The top $\pi = p\mu$ agents in the population are kept, and the remaining $\mu - \pi$ are generated from scratch.

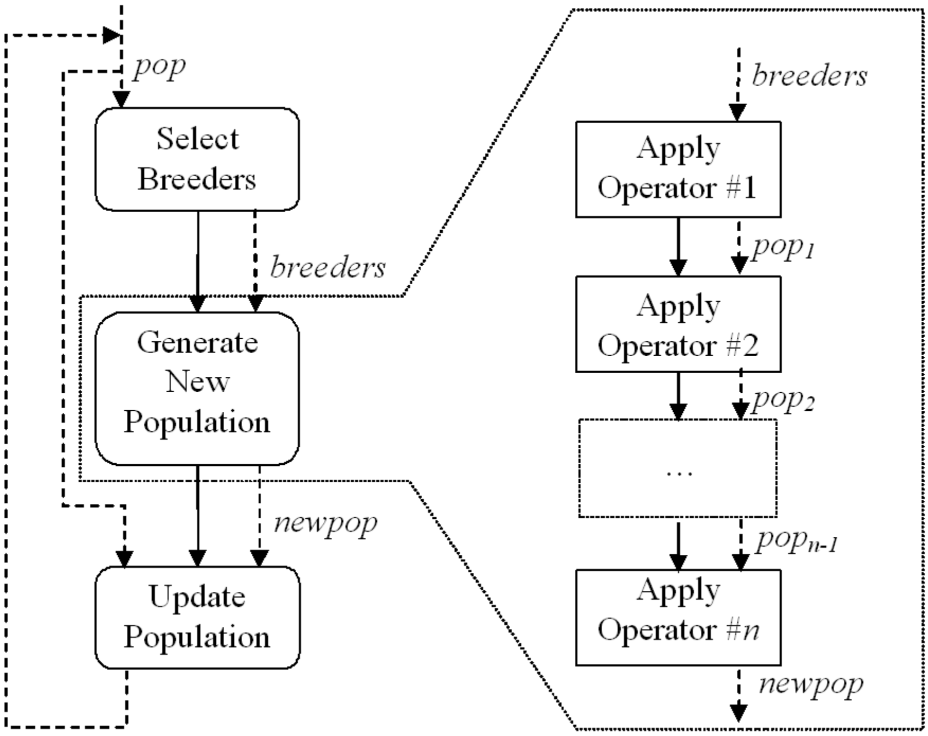

Figure 1.4: The basic generational step. Notice the use of a pipeline of reproductive operators for creating new solutions.

Table 1.1: Some applications of memetic algorithms in combinatorial optimization.

Graph Partitioning	[13][70]	Min Number Partitioning	[9][10]
Max Independent Set	[2][40][93]	Bin-Packing	[90]
Min Graph Coloring	[20][33]	Set Covering	[7]
Min Generalized Assignment	[19]	Multidimensional Knapsack	[25][49][100]
Quadratic Assignment	[69][71]	Quadratic Programming	[72]
Set Partitioning	[59]	Gate Matrix Layout	[63][64]
Traveling Salesman Problem	[14][42][56]	Min Weighted k-Cardinality Tree	[12]
	[67][91][107]	Min k-Cut Problem	[105]
Uncapacitated Hub Location	[1]	Placement Problems	[43][58][95]
Vehicle Routing	[8][48][85]	Task Allocation	[39]
Prize-Collecting Steiner Tree	[53]	Network Design	[4][86][92]
Vertex-Biconnectivity Augmentation	[51]	Error Correcting Codes	[23]
OSPF routing	[15]	Maintenance Scheduling	[16]
Open Shop Scheduling	[18]	Flowshop Scheduling	[36][46][98]
Single Machine Scheduling	[37][62][65]	Parallel Machine Scheduling	[66]
Project Scheduling	[88]	Production Planning	[32]
Timetabling	[3][84]	Rostering	[31]
Sport Games Scheduling	[21][96]	Airport Gate Scheduling	[60][106]
Multistage Capacitated Lot-Sizing	[11]	Graph Isomorphism Problem	[103]
Protein Structure Prediction	[5][6][57]	Clustering	[68][83][101]

Index