1

Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews

C. Grosan and A. Abraham

Summary. Evolutionary computation has become an important problem solving methodology among many researchers. The population-based collective learning process, self-adaptation, and robustness are some of the key features of evolutionary algorithms when compared to other global optimization techniques. Even though evolutionary computation has been widely accepted for solving several important practical applications in engineering, business, commerce, etc., yet in practice sometimes they deliver only marginal performance. Inappropriate selection of various parameters, representation, etc. are frequently blamed. There is little reason to expect that one can find a uniformly best algorithm for solving all optimization problems. This is in accordance with the No Free Lunch theorem, which explains that for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class. Evolutionary algorithm behavior is determined by the exploitation and exploration relationship kept throughout the run. All these clearly illustrates the need for hybrid evolutionary approaches where the main task is to optimize the performance of the direct evolutionary approach. Recently, hybridization of evolutionary algorithms is getting popular due to their capabilities in handling several real world problems involving complexity, noisy environment, imprecision, uncertainty, and vagueness. In this chapter, first we emphasize the need for hybrid evolutionary algorithms and then we illustrate the various possibilities for hybridization of an evolutionary algorithm and also present some of the generic hybrid evolutionary architectures that has evolved during the last couple of decades. We also provide a review of some of the interesting hybrid frameworks reported in the literature.

1.1 Introduction

Evolutionary computation, offers practical advantages to the researcher facing difficult optimization problems. These advantages are multifold, including the simplicity of the approach, its robust response to changing circumstance, its flexibility, and many other facets. The evolutionary algorithm can be applied to problems where heuristic solutions are not available or generally lead to unsatisfactory results. As a result, evolutionary algorithms have recently received increased interest, particularly with regard to the manner in which they may be applied for practical

C. Grosan and A. Abraham: *Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews*, Studies in Computational Intelligence (SCI) **75**, 1–17 (2007)
www.springerlink.com

problem solving. Usually grouped under the term evolutionary computation or evolutionary algorithms, we find the domains of genetic algorithms [23], evolution strategies [56], [58], evolutionary programming [15], and genetic programming [31]. They all share a common conceptual base of simulating the evolution of individual structures via processes of selection, mutation, and reproduction. The processes depend on the perceived performance of the individual structures as defined by the problem. Compared to other global optimization techniques, evolutionary algorithms (EA) are easy to implement and very often they provide adequate solutions. The flow chart of an EA is illustrated in Fig. 1.1. A population of candidate solutions (for the optimization task to be solved) is initialized. New solutions are created by applying reproduction operators (mutation and/or crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions are to be maintained into the next generation. The procedure is then iterated.

For several problems a simple Evolutionary algorithm might be good enough to find the desired solution. As reported in the literature, there are several types of problems where a direct evolutionary algorithm could fail to obtain a convenient (optimal) solution [37,40,61,65]. This clearly paves way to the need for hybridization of evolutionary algorithms with other optimization algorithms, machine learning techniques, heuristics etc. Some of the possible reasons for hybridization are as follows [60]:

1. To improve the performance of the evolutionary algorithm (example: speed of convergence)
2. To improve the quality of the solutions obtained by the evolutionary algorithm
3. To incorporate the evolutionary algorithm as part of a larger system

In 1995, Wolpert and Macready [73] illustrated that all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions. According to the authors, if algorithm *A* outperforms algorithm *B* on some cost functions, then loosely speaking there must exist exactly

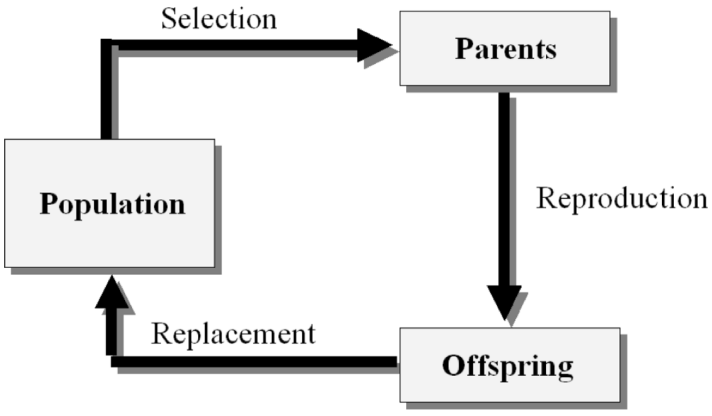

Fig. 1.1. Flowchart of an evolutionary algorithm

as many other functions where *B* outperforms *A*. Hence, from a problem solving perspective it is difficult to formulate a universal optimization algorithm that could solve all the problems. Hybridization may be the key to solve practical problems. To illustrate the popularity of hybrid approaches, we searched the number of publications appearing in some of the popular scientific databases namely ScienceDirect [74], IEEE-Xplore [76], and SpringerLink [75] using the keywords "hybrid evolutionary" and "hybrid genetic" and the query results are tabulated below. Since no filtering was used in the query, the number of relevant papers might be lower than the figures mentioned.

Keyword	Science Direct	IEEE Explore	SpringerLink
hybrid evolutionary	4,674	120	535
hybrid genetic	5,614	296	6,158

Figure 1.2 illustrates some possibilities for hybridization. From initialization of population to the generation of offsprings, there are lots of opportunities to incorporate other techniques/algorithms etc. Population may be initialized by incorporating known solutions or by using heuristics, local search etc. Local search methods may be incorporated within the initial population members or among the offsprings. Evolutionary algorithms may be hybridized by using operators from

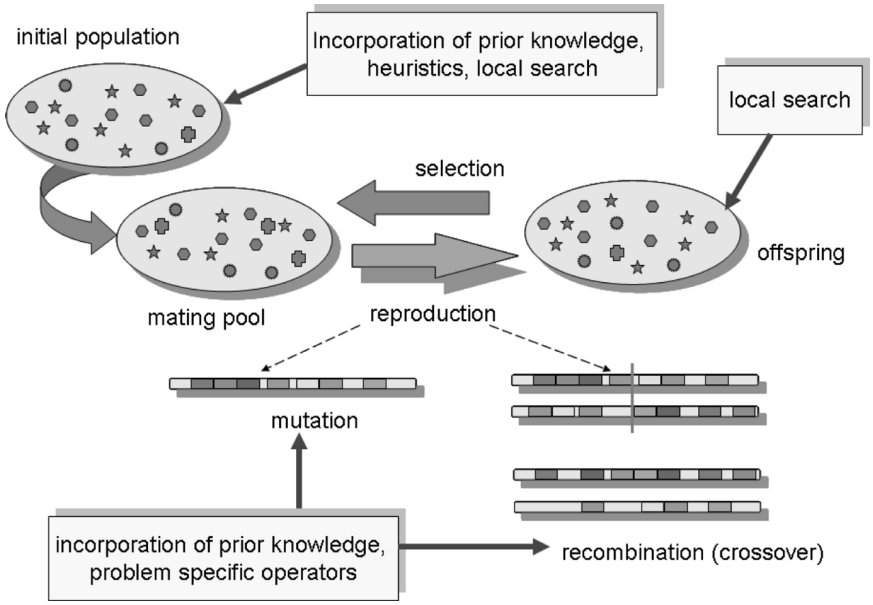

Fig. 1.2. Hybridization prospectives in an evolutionary algorithm

other algorithms (or algorithms themselves) or by incorporating domain-specific knowledge. Evolutionary algorithm behavior is determined by the exploitation and exploration relationship kept throughout the run. Adaptive evolutionary algorithms have been built for inducing exploitation/exploration relationships that avoid the premature convergence problem and optimize the final results. The performances of the evolutionary algorithm can be improved by combining problem-specific knowledge for particular problems.

The rest of the chapter is organized as follows. In Sect. 1.2, the various architectures for hybrid evolutionary algorithms are presented. In Sect. 1.3, we review the different hybrid evolutionary algorithms and some conclusions are provided toward the end.

1.2 Architectures of Hybrid Evolutionary Algorithms

As reported in the literature, several techniques and heuristics/metaheuristics have been used to improve the general efficiency of the evolutionary algorithm. Some of most used hybrid architectures are summarized as follows:

1. Hybridization between an evolutionary algorithm and another evolutionary algorithm (example: a genetic programming technique is used to improve the performance of a genetic algorithm)
2. Neural network assisted evolutionary algorithms
3. Fuzzy logic assisted evolutionary algorithm
4. Particle swarm optimization (PSO) assisted evolutionary algorithm
5. Ant colony optimization (ACO) assisted evolutionary algorithm
6. Bacterial foraging optimization assisted evolutionary algorithm
7. Hybridization between evolutionary algorithm and other heuristics (such as local search, tabu search, simulated annealing, hill climbing, dynamic programming, greedy random adaptive search procedure, etc)

In the following sections, we will briefly review some of the architectures depicted above. Figure 1.3 illustrates some of the generic architectures for the various types of hybridization. By problem, we refer to any optimization or even function approximation type problem and intelligent paradigm refers to any computational intelligence technique, local search, optimization algorithms etc.

Figure 1.3a, b represents a concurrent architecture where all the components are required for the proper functioning of the model. As depicted in Fig. 1.3a, evolutionary algorithm acts as a preprocessor and the intelligent paradigm is used to fine tune the solutions formulated by the evolutionary algorithm. In Fig. 1.3b, intelligent paradigm acts as a preprocessor and the evolutionary algorithm is used to fine tune the solutions formulated by the intelligent paradigm. Figure 1.3c, represents a transformational hybrid system in which the evolutionary algorithm is used to fine tune the performance of the intelligent paradigm and at the same time, the intelligent paradigm is used to optimize the performance of the evolutionary algorithm. Required information is exchanged between the two techniques during the search

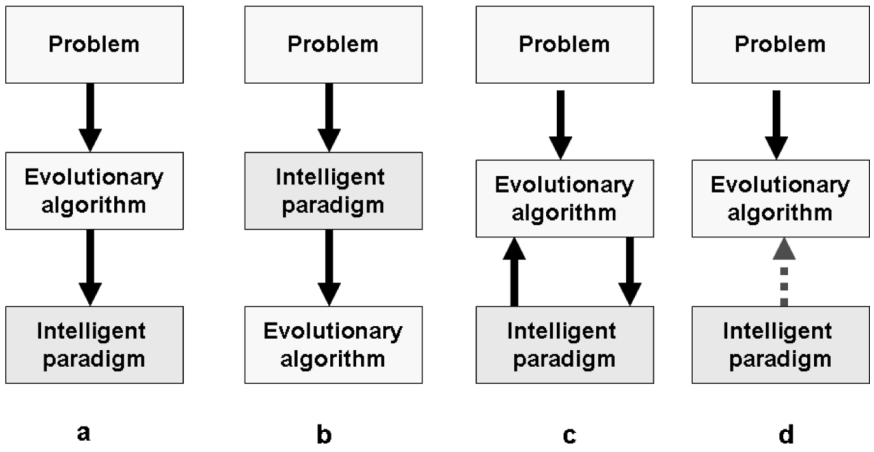

Fig. 1.3. Hybrid evolutionary algorithm generic architectures

(problem solving) process. In a cooperative model the intelligent paradigm is used only for initialization or for determining some parameters of the evolutionary algorithm. As depicted in Fig. 1.3d, thereafter, the intelligent paradigm is not required for the proper functioning of the system. Also, there are several ways to hybridize two or more techniques.

In Sect. 1.3, some of the well established hybrid frameworks for optimizing the performance of evolutionary algorithm using intelligent paradigms are presented.

1.3 Hybrid Evolutionary Architectures

The integration of different learning and adaptation techniques, to overcome individual limitations and achieve synergetic effects through hybridization or fusion of these techniques, has in recent years contributed to a large number of new hybrid evolutionary systems. Most of these approaches, however, follow an ad hoc design methodology, further justified by success in certain application domains. Due to the lack of a common framework it remains often difficult to compare the various hybrid systems conceptually and evaluate their performance comparatively. There are several ways to hybridize a conventional evolutionary algorithm for solving optimization problems. Some of them are summarized below [63]:

- The solutions of the initial population of EA may be created by problem-specific heuristics.
- Some or all the solutions obtained by the EA may be improved by local search. This kind of algorithms are known as memetic algorithms [21, 50].
- Solutions may be represented in an indirect way and a decoding algorithm maps any genotype to a corresponding phenotypic solution. In this mapping, the decoder can exploit problem-specific characteristics and apply heuristics etc.

– Variation operators may exploit problem knowledge. For example, in recombination more promising properties of one parent solution may be inherited with higher probabilities than the corresponding properties of the other parent(s). Also mutation may be biased to include in solutions promising properties with higher probabilities than others.

1.3.1 Evolutionary Algorithms Assisted by Evolutionary Algorithms

Tan et al. [64] proposed a two-phase hybrid evolutionary classification technique to extract classification rules that can be used in clinical practice for better understanding and prevention of unwanted medical events. In the first phase, a hybrid evolutionary algorithm is used to confine the search space by evolving a pool of good candidate rules. Genetic programming [32] is applied to evolve nominal attributes for free structured rules and genetic algorithm is used to optimize the numeric attributes for concise classification rules without the need of discretization. These candidate rules are then used in the second phase to optimize the order and number of rules in the evolution for forming accurate and comprehensible rule sets.

Zmuda et al. [69] proposed an hybrid evolutionary learning scheme for synthesizing multiclass pattern recognition systems. A considerable effort is spent for developing complex features that serve as inputs to a simple classifier back end. The nonlinear features are created using a combination of genetic programming [32–35] to synthesize arithmetic expressions, genetic algorithms [23] to select a viable set of expressions, and evolutionary programming [13, 14] to optimize parameters within the expressions. The goal is create a compact set of nonlinear features that cooperate to solve a multiclass pattern recognition problem.

Swain and Morris /citeswain proposed an hybridization between evolutionary programming (EP) and a fitness-blind mutation (FBM) algorithm. The method developed by the authors is functionally, and structurally equivalent to standard EP, but still can be used effectively to optimize functions having strong fitness dependency between parents and their offspring. The FBM algorithm is used in conjunction with the EP mutation operator. The FBM operation has been implemented by taking the standard deviation of the Gaussian variable to vary in proportion to the genotypic distance between the individual parent and the fittest individual, which is defined as a pseudoglobal optimum individual in a population pool. Also, the directionality of the random variation has been exploited to improve the probability of getting better solutions. In addition to this, the importance of initial search width for generating the offspring has been established empirically.

1.3.2 Evolutionary Algorithms Assisted by Neural Networks

Wang [71] proposed a hybrid approach to improve the performance of evolutionary algorithms for a simulation optimization problem. Simulation optimization aims at determining the best values of input parameters, while the analytical objective function and constraints are not explicitly known in terms of design variables and their values only can be estimated by complicated analysis or time-consuming simulation [71].

In the first phase, neural networks (NN) are constructed based on a collection of training samples. Then, the evolutionary algorithm is used to explore good solutions among the solution space. Once the evolutionary algorithm generates a new solution, the NN will be used to determine its fitness value for the evolutionary algorithm to continue its search process. Until the stopping criterion of the evolutionary algorithm is satisfied, the strategy will output the best solution resulted by the evolutionary algorithm and its performance determine by detailed evaluation based on actual problem.

To improve the consistency and robustness of the results, it is suggested to use multiple NN's to provide statistical predicted performance for the evolutionary algorithm. For those problems with known form of objective function but hard to evaluate the performance, NN still can be established to rapidly provide performance evaluation to enhance the efficiency of genetic search.

1.3.3 Fuzzy Logic Assisted Evolutionary Algorithms

Fuzzy logic controller (FLC) is composed by a knowledge base, that includes the information given by the expert in the form of linguistic control rules, a fuzzification interface, which has the effect of transforming crisp data into fuzzy sets, an inference system, that uses them together with the knowledge base to make inference by means of a reasoning method, and a defuzzification interface, that translates the fuzzy control action thus obtained to a real control action using a defuzzification method. FLCs have been used to design adaptive evolutionary algorithms. The main idea is to use an FLC whose inputs are any combination of EA performance measures and current control parameter values and whose outputs are EA control parameter values. Lee and Takagi [36] proposed the dynamic parametric GA (DPGA) that uses an FLC for controlling GA parameters. The inputs to the FLC are any combination of GA performance measures or current control settings, and outputs may be any of the GA control parameters. Hererra and Lozano [22] reported tightly coupled, uncoupled, and loosely coupled methods for adaptation. Three levels of tightly coupled adaptation may be implemented at the level of individuals, the level of subpopulations and the level of population. In an uncoupled adaptation, a totally separate adaptive mechanism adjusts the performance of EA. It is to be noted that an uncoupled approach does not rely upon the EA for the adaptive mechanism. In the loosely coupled method, EA is partially used for the adaptive mechanism, i.e., either the population or the genetic operators are used in some fashion.

The EA control parameter settings such as mutation probability (P_{m}), crossover probability (P_{c}), and population size (N) are key factors in the determination of the exploitation versus exploration tradeoff.

Example: Mutation rates (P_{m}) may be adapted to prevent premature convergence and to speed up the optimization. The rules that take care of adjusting mutation rates could be formulated as follows:

- If convergent then set $P_{\mathrm{m}} = 0.6$
- If not convergent then set $P_{\mathrm{m}} = 0.05$

1.3.4 Evolutionary Algorithms Assisted by Particle Swarm Optimization

PSO incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the idea is emerged [7, 9, 27, 28].

A hybrid evolutionary algorithm – PSO method is proposed by Shi et al. [59]. The hybrid approach executes the two systems simultaneously and selects P individuals from each system for exchanging after the designated N iterations. The individual with larger fitness has more opportunities of being selected. The main steps of the hybrid approach are depicted below [59]:

1. Initialize EA and PSO subsystems.
2. Execute EA and PSO simultaneously.
3. Memorize the best solution as the final solution and stop if the best individual in one of the two subsystems satisfies the termination criterion.
4. Perform the hybrid process if generations could be divided exactly by the designated number of iterations N. Select P individuals from both sub-systems randomly according to their fitness and exchange. Go to step 3.

A hybrid technique combining GA and PSO called genetic swarm optimization (GSO) is proposed by Grimaldi et al. [18] for solving an electromagnetic optimization problem. The method consists of a strong co-operation of GA and PSO, since it maintains the integration of the two techniques for the entire run. In each iteration, the population is divided into two parts and they are evolved with the two techniques, respectively. They are then recombined in the updated population, that is again divided randomly into two parts in the next iteration for another run of genetic or particle swarm operators. The population update concept can be easily understood thinking that a part of the individuals is substituted by new generated ones by means of GA, while the remaining are the same of the previous generation but moved on the solution space by PSO.

The driving parameter of the PSO algorithm is the hybridization coefficient (HC), which expresses the percentage of population that in each iteration is evolved with GA. So HC = 0 means the procedure is a pure PSO (the whole population is updated according to PSO operators), HC = 1 means pure GA, while $0 <$HC< 1 means that the corresponding percentage of the population is developed by GA and the rest by PSO.

Grosan et al. [19] proposed a variant of the PSO technique named independent neighborhoods particle swarm optimization (INPSO) dealing with subswarms for solving the well known geometrical place problems. The performance of the INPSO approach is compared with Geometrical Place Evolutionary Algorithms (GPEA). The main advantage of the INPSO technique is its speed of convergence (finding quick solutions). To enhance the performance of the INPSO approach, a hybrid algorithm combining INPSO and GPEA is also proposed in this paper. The developed hybrid combination is able to detect the geometrical place much faster even for difficult problems for which the direct GPEA approach required more time and the

INPSO (even with few subswarms) approach failed in finding all the geometrical place points (solutions).

Liu et al. [42] introduced turbulence in the particle swarm optimization (TPSO) algorithm to overcome the problem of stagnation. The algorithm used a minimum velocity threshold to control the velocity of particles. TPSO mechanism is similar to a turbulence pump, which supplies some power to the swarm system to explore new neighborhoods for better solutions. The algorithm also avoids clustering of particles and at the same time attempts to maintain diversity of population. The parameter, the minimum velocity threshold of the particles is tuned adaptively by the FLCs in the TPSO algorithm, which is further called as Fuzzy Adaptive TPSO (FATPSO). The comparison was performed on a suite of 20 widely used benchmark problems.

1.3.5 Evolutionary Algorithms Assisted by Ant Colony Optimization

ACO deals with artificial systems that are inspired from the foraging behavior of real ants, which are used to solve discrete optimization problems [8].

Tseng and Liang [65] proposed a hybrid approach that combines (ACO), the genetic algorithm (GA) and a Local Search (LS) method. The algorithm is applied for solving the Quadratic Assignment Problem (QAP). Instead of starting from a population that consists of randomly generated chromosomes, GA has an initial population constructed by ACO in order to provide a good start. Pheromone acts as a feedback mechanism from GA phase to ACO phase. When GA phase reaches the termination criterion, control is transferred back to ACO phase. Then ACO utilizes pheromone updated by GA phase to explore solution space and produces a promising population for the next run of GA phase. The local search method is applied to improve the solutions obtained by ACO and GA. Another hybrid approach for the same problem were proposed by Vasquez and Whitley [67] where GA is combined with Tabu Search.

Ahuja et al. [2] used a greedy genetic algorithm. This approach incorporates new ideas such as: generating the initial population using a randomized construction heuristic; new crossover schemes; a special purpose immigration scheme that promotes diversity; periodic local optimization of a subset of the population; formulating tournaments among different populations; and an overall design that attempts to strike a balance between diversity and a bias toward better individuals. Fleurent and Ferland proposed some general hybrid approaches for combining genetic algorithms and heuristics for QAP [12]. Genetic algorithms hybridized with repair and recreate procedure are applied for QAP by Misevicius [48].

1.3.6 Evolutionary Algorithms Assisted by Bacterial Foraging

Recently search and optimal foraging decision-making of bacteria has been used for solving optimization problems [55]. The foraging behavior of *Escherichia coli* bacteria is mimicked. They undergo different stages such as chemotaxis, swarming, reproduction, and elimination and dispersal. In the chemotaxis stage, it can have tumble followed by a tumble or a tumble followed by a run. On the other hand, in swarming, each *E. coli* bacterium will signal other via attractants to swarm together.

In elimination and dispersal, any one bacterium is eliminated from the total set just by dispersing it to a random location on the optimization domain.

Kim et al. [29] used a hybrid genetic algorithm (HGA) and bacterial foraging approach for function optimization and PID controller tuning. In the hybrid framework, the population members may be viewed as a group of bacteria foraging in the problem search space. Experiment results clearly reveal that the hybrid approach performed better than a direct GA approach.

1.3.7 Evolutionary Algorithms Incorporating Prior Knowledge

There are several existing approaches which are not using a randomly generated initial population for evolutionary algorithms. If prior knowledge exists or can be generated at a low computational cost, with good initial estimates may generate better solutions with faster convergence [20, 24, 38, 52, 70].

Keedwell and Khu [26] mentioned that a heuristic-based approach to seeding a GA should yield performance enhancements on difficult problems. Authors [26] proposed a heuristic-based local representative cellular automata (CA) [53] approach to provide a good initial population for evolutionary algorithm runs. Grefenstette [17] discussed methods and demonstrated the value of incorporating problem-specific knowledge into the EA mechanism, including seeding the population. Louis [41] seeded the EA population with known good solutions from case-based reasoning. The approach was tested for the open shop rescheduling problem and found that the performance of EA was consistently better than a randomly seeded EA. Oman and Cunningham [54] experimented with seeding for the Traveling Salesman Problem (TSP) and the job-shop scheduling problem (JSSP), as two benchmark tasks for evolutionary algorithms. They seeded the EA with known good solutions in the initial population and found that the results were significantly improved on the TSP but not on JSSP. Interestingly, they used a varying percentage of seeding, from 25 to 75% and the result for each was remarkably similar although the authors pointed out that a 100% seed was not very successful on either problems [26].

1.3.8 Hybrid Approaches Incorporating Local Search and Others

Hybridization between evolutionary algorithms and local search is known as memetic algorithms. Memetic Algorithms have been proved to be orders of magnitude faster and more accurate than evolutionary algorithms for different classes of problems. As reported in the literature, hybrid methods combining probabilistic methods and deterministic methods have found success in solving complex optimization problems [4–6].

Evolutionary programming (EP) hybridized with a deterministic optimization procedure was applied by Myung and Kim [51] for nonlinear and quadratic optimization problems. The hybrid approach is outperforming EP alone, two-phase (TP) optimization, and EP with a (NS-EP) in terms of computational efficiency and solution accuracy.

Somasundaram et al. [61] proposed a hybrid method for solving the security constrained economic dispatch problem [66]. A simple evolutionary programming is applied as a base level search, which can give a good direction to the optimal global region and a local search linear programming (LP) is used to fine tune to determine the optimal solution. For the problem considered, authors found [61] that initially, the rate of convergence in EP is very fast and subsequently the convergence is very slow. A direct LP approach will be effective only when the magnitudes of constraint violations are less and corrections in the control variables are small. In order to overcome these difficulties, a two-phase hybrid method has been proposed by the authors [61].

A HGA for permutation flow shop with limited buffers was proposed by Wang et al. [72]. Multiple genetic operators are applied simultaneously in a hybrid sense to perform evolutionary search (globally) and a local search based on a neighborhood structure-based graph model. The utilization of genetic mutation and local search is controlled by a decision probability so that population diversity can be maintained and computing effort can be concentrated on promising neighbor solutions. Elitism is also used so that the best solution found so far cannot be lost [72].

Burke and Smith [6] proposed a hybrid EA-local search for the thermal generator maintenance scheduling problem. A heuristic is used for solutions initialization. Fatourechi et al. [11] used a HGA for user customization of the energy normalization parameters in brain–computer interface systems. The GA is hybridized with a local search algorithm in their approach. Schlottmann and Seese [57] proposed a hybrid heuristic approach combining multiobjective evolutionary and problem-specific local search methods to support the risk-return analysis of credit portfolios. Menon et al. [46] used two hybrid techniques combining differential evolution [62] and local search for the clearance of nonlinear flight control laws.

Estudillo et al. [10] proposed a combination of an EA, a clustering process, and a local-search procedure for the evolutionary design of neural networks [1]. The local-search method is incorporated into the EA in order to improve its performance. In order to efficiently use the hybrid algorithm, it is not worth to carry out a local optimization algorithm for every individual in the population due to the size of the population and/or the dimension of the search space. The proposed approach selects a subset of the best individuals, perform a cluster analysis to group them, and optimize only the best individual of every group. The use of a clustering algorithm allows the selection of individuals representing different regions in the search space. In this way, the optimized individuals are more likely to converge toward different local optima.

Aruldoss and Ebenezer [3, 4] proposed a – (SQP) method for the dynamic economic dispatch problem (DEDP) of generating units considering the valve-point effects. The developed method is a two-phase optimizer. In the first phase, the candidates of EP explores the solution space freely. In the second phase, the SQP is invoked when there is an improvement of solution (a feasible solution) during the EP run. Thus, the SQP guides EP for better performance in the complex solution space. A similar hybrid approach involving EP and SQP techniques was proposed by Attaviriyanupap et al. [5], where the EP is applied to obtain a near global solution;

once the EP terminates its procedure, the SQP is applied to obtain final optimal solution.

Lin et al. [39] proposed a hybrid approach to deal with the mixed-integer optimization problems. The hybrid algorithm contains the migration operation to avoid candidate individuals clustering together. The population diversity measure is introduced in order to inspect when the migration operation should be performed so that a smaller population size can be used to obtain a global solution. A mixed coding representation and a rounding operation are introduced. A migration operation is embedded in the algorithm so that it is able to obtain a global solution using a small population size.

Jeong et al. [25] suggested an hybrid approach with a genetic algorithm (GA) and a simulation technique. The GA is used for optimization of schedules, and the simulation is used to minimize the maximum completion time for the last job with fixed schedules obtained from the GA model. The main steps of the approach are [25]:

1. Using GA and generate production schedules
2. Run a simulation model based on the GA generated production schedules
3. Obtain feasible simulation completion time
4. Decide on the appropriate result, which yields the required values
5. Change constraints in the GA using current simulation completion time and go to step 1
6. Determine the production scheduling which is considered to be the realistic optimal solution

Ganesh and Punniyamoorthy [16] proposed a hybrid GA – simulated annealing (SA) algorithm for continuous-time aggregate production-planning problems. The motivation behind the GA–SA combination is the power of GA to work on the solution in a global sense while allowing SA to locally optimize each individual solution [16]. The hybrid algorithm executes in two phases. In the first phase, the GA generates the initial solutions randomly. The GA then operates on the solutions using selection, crossover, and mutation operators to produce new and hopefully better solutions. After each generation, the GA sends each solution to the SA (second phase) to be improved. The neighborhood generation scheme used in SA is a single insertion neighborhood scheme. Once the SA is finished for a solution of GA, another GA solution is passed to SA. This process continues until all solutions of GA in one generation are exhausted. Once the SA is finished for all solutions in one generation of GA, the best solutions of population size obtained from SA are the solutions of GA for the next generation. The GA and SA exchange continues until the required number of generations are completed [16].

Kim et al. [30] hybridized a modified EP with subsequent deterministic optimization following a Lagrange multiplier method [43]. The obtained approach is applied for constraint optimization problems.

Arc revision [44] is a method of constraint processing which removes from the domain of the variable at the tail of an arc any value which is not supported through the arc by at least one value in the domain of the variable at the head of the arc.

Dozier et al. [8] used the conventional evolutionary search with the systematic search concepts of arc revision and two variants of hill climbing [47, 49] to form a hybrid system that quickly finds solutions to static and dynamic constraint satisfaction problems.

Lo and Chang [40] proposed a for the capacitated multipoint network design problem. The concept of subpopulations is used. Four subpopulations are generated according to the elitism reservation strategy, the shifting Prüfer vector, the stochastic universal sampling, and the complete random method, respectively. Mixing these four subpopulations produces the next generation population. Magyar et al. [45] proposed a HGA with an adaptive application of genetic operators for solving the (3MP). Several general/heuristic crossover and local for the 3MP, and adaptation is applied at the level of choosing among the operators. The GA combines these operators to form an effective problem solver. The algorithm is hybridized as it incorporates local search heuristics, and it is adaptive as the individual recombination/improvement operators are fired according to their on-line performance.

Zhong and Yang proposed a HGA to solve the tasks scheduling problem [68]. It uses genetic algorithm to evolve tasks dispatching priority queue, and uses list scheduling to decode the queue info a schedule. In order to remedy the GA's weakness in fine-tuning, a neighborhood search method is used to improve the fitness of the individuals of each generation, based on Lamarckian theory of evolution.

1.4 Conclusions

As evident from the scientific literature/databases, the use of hybrid evolutionary algorithms are getting very popular. In this chapter, we illustrated the various possibilities for hybridization of an evolutionary algorithm and also presented some of the generic hybrid evolutionary architectures that has evolved during the last couple of decades. We also provided a review of some of the interesting hybrid frameworks reported in the literature.

References

1. Abraham A (2004) Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, 56c, pp. 1–38
2. Ahuja RK, Orlin JB, and Tiwari A (2000) A greedy genetic algorithm for the quadratic assignment problem, Computers and Operations Research, 27, 917–934
3. Aruldoss AVT and Ebenezer JA (2004) Hybrid PSO-SQP for economic dispatch with valve-point effect, Electric Power Systems Research, 71(1), pp. 51–59
4. Aruldoss AVT and Ebenezer JA (2005) A modified hybrid EP-SQP approach for dynamic dispatch with valve-point effect, International Journal of Electrical Power and Energy Systems, 27(8), pp. 594–601
5. Attaviriyanupap KH, Tanaka E, and Hasegawa J (2002) A hybrid EP and SQP for dynamic economic dispatch with nonsmooth incremental fuel cost function, IEEE Transaction on Power Systems, 17(2), pp. 411–416

6. Burke EK and Smith AJ (2000) Hybrid evolutionary techniques for the maintenance scheduling problem, IEEE Transactions on Power Systems, 1(1), pp. 122–128
7. Clerc M and Kennedy J (2002) The particle swarm: explosion stability and convergence in a multi-dimensional complex space, IEEE Transaction on Evolutionary Computation 6(1), pp. 58–73
8. Dozier G, Bowen J, and Homaifar A (1998) Solving constraint satisfaction problems using hybrid evolutionary search, IEEE Transactions on Evolutionary Computation, 2(1), pp. 23–33
9. Eberhart RC and Kennedy J (1995) A new optimizer using particle swarm theory, In Proceedings of 6th Internationl Symposium on Micro Machine and Human Science, Nagoya, Japan, IEEE Service Center, Piscataway, NJ, pp. 39–43
10. Estudillo ACM, Martínez CH, Estudillo FJM, and Pedrajas GC (2006) Hybridization of evolutionary algorithms and local search by means of a clustering method, IEEE Transactions on Systems, Man and Cybernetics, Part B, 36(3), pp. 534–545
11. Fatourechi M, Bashashati A, Ward RK, and Birch G (2005) A hybrid genetic algorithm approach for improving the performance of the LF-ASD brain computer interface, In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP05), Philadelphia, pp. 345–348
12. Fleurent C and Ferland J (1994) Genetic hybrids for the quadratic assignment problems, In Quadratic Assignment and Related Problems, Pardalos PM and Wolkowicz H (Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16, AMS: Providence, Rhode Island, pp. 190–206
13. Fogel DB (1991) System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling, Ginn & Co., Needham, MA
14. Fogel DB (1994) An introduction to simulated evolutionary optimization. IEEE Transaction on Neural Networks, 5(1), pp. 3–14
15. Fogel LJ, Owens AJ, and Walsh MJ (1966) Artificial Intelligence Through Simulated Evolution, Wiley, USA
16. Ganesh K and Punniyamoorthy M (2004) Optimization of continuous-time production planning using hybrid genetic algorithms-simulated annealing, International Journal of Advanced Manufacturing Technology, 26(1), pp. 148–154
17. Grefenstette JJ (1987) Incorporating problem specific knowledge into genetic algorithms, In Davis L. (Ed.), Genetic Algorithms and Simulated Annealing, Morgan Kaufmann, Los Altos, CA, pp. 42–60
18. Grimaldi EA, Grimacia F, Mussetta M, Pirinoli P, and Zich RE (2004) A new hybrid genetical – swarm algorithm for electromagnetic optimization, In Proceedings of International Conference on Computational Electromagnetics and its Applications, Beijing, China, pp. 157–160
19. Grosan C, Abraham A, and Nicoara M (2005) Search optimization using hybrid particle sub-swarms and evolutionary algorithms, International Journal of Simulation Systems, Science and Technology, UK, 6(10–11), pp. 60–79
20. Harik GR and Goldberg DE (2000) Linkage learning through probabilistic expression. Computer Methods in Applied Mechanics and Engineering 186(2–4), pp. 295–310
21. Hart WE, Krasnogor N, and Smith JE (Eds.) (2005) Recent Advances in Memetic Algorithms, Series: Studies in Fuzziness and Soft Computing, Vol. 166
22. Herrera F and Lozano M (1996) Adaptation of genetic algorithm parameters based on Fuzzy logic controllers. Genetic Algorithms and Soft Computing, Herrera F, Verdegay JL (Eds.), pp. 95–125
23. Holland JH (1975) Adaptation in natural and artificial systems, The University of Michigan Press, Ann Arbor, MI

24. Hopper E and Turton BCH (2001) An empirical investigation of metaheuristic and heuristic algorithms for a 2D packing problem, European Journal of Operational Research 128, pp. 34–57
25. Jeong SJ, Lim SJ, and Kim KS (2005) Hybrid approach to production scheduling using genetic algorithm and simulation, International Journal of Advanced Manufacturing Technology, 28(1), pp. 129–136
26. Keedwell E and Khu ST (2005) A hybrid genetic algorithm for the design of water distribution networks, Engineering Applications of Artificial Intelligence, 18(4), pp. 461–472
27. Kennedy J and Eberhart RC (1995) Particle swarm optimization, In Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948
28. Kennedy J (1997) The particle swarm: social adaptation of knowledge, In Proceedings of IEEE International Conference on Evolutionary Computation, Indianapolis, IN, 1997, pp. 303–308
29. Kim DH and Cho JH (2005) Robust tuning of PID controller using bacterial-foraging-based optimization, JACIII 9(6), 669–676
30. Kim JH, Myung H, and Jeon JY (1995) Hybrid evolutionary programming with fast convergence for constrained optimization problems, IEEE Conference on Intelligent Systems for the 21 Century, Vancouver, Canada, pp. 3047–3052
31. Koza JR (1992) Genetic Programming, MIT Press, Cambridge, MA
32. Koza JR (1992) Genetic Programming: On the Programming of Computers By Means of Natural Selection, MIT Press, Cambridge, MA
33. Koza JR (1994) Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, Cambridge, MA
34. Koza JR, Bennett FH, Andre D, and Keane MA (1999) Genetic programming III: Darwinian invention and problem solving, Morgan Kaufmann, Los Altos, CA
35. Koza JR, Keane MA, Streeter MJ, Mydlowec W, Yu J, and Lanza G (2003) Genetic programming IV: Routine human-competitive machine intelligence, Kluwer, Dordecht
36. Lee MA and Takagi H (1993) Dynamic control of genetic algorithms using fuzzy logic techniques, In Forrest S (Ed.), Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmmann, San Mateo, pp 76–83
37. Li F, Morgan R, and Williams D (1997) Hybrid genetic approaches to ramping rate constrained dynamic economic dispatch, Electric Power Systems Research, 43(11), pp. 97–103
38. Liaw CF (2000) A hybrid genetic algorithm for the open shop scheduling problem, European Journal of Operational Research 124(1), pp. 28–42
39. Lin YC, Hwang KS, and Wang FS (2004) A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems, Computers and Mathematics with Applications, 47(8–9), pp. 1295–1307
40. Lo CC and Chang WH (2000) A multiobjective hybrid genetic algorithm for the capacitated multipoint network design problem, IEEE Transactions on Systems, Man and Cybernetics - Part B, 30(3), pp. 461–470
41. Louis SJ (1997) Working from blueprints: evolutionary learning for design, Artificial Intelligence in Engineering, 11, pp. 335–341
42. Liu H, Abraham A, and Zhang W (2007) A Fuzzy adaptive turbulent particle swarm optimization, International Journal of Innovative Computing and Applications, 1(1), pp. 39–47
43. Maa CY and Shanblatt MA (1992) A two-phase optimization neural network, IEEE Transaction on Neural Networks, 3(6), pp. 1003–1009
44. Mackworth AK (1977) Consistency in networks of relations, Artificial Intelligence, 8, pp. 98–118

45. Magyar G, Johnsson M, and Nevalainen O (2000) An adaptive hybrid genetic algorithm for the three-matching problem, IEEE Transactions on Evolutionary Computation, 4(2), pp. 135–146
46. Menon PP, Bates DG, and Postlethwaite I (2005) Hybrid evolutionary optimisation methods for the clearance of nonlinear flight control laws, In Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, Seville, Spain, pp. 4053–4058
47. Minton S, Johnston MD, Philips AB, and Laird P (1992) Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems, Artificial Intelligence, 58, pp. 161–205
48. Misevicius A (2003) Genetic algorithm hybridized with ruin and recreate procedure: Application to the quadratic assignment problem, Knowledge-Based Systems, 16, pp. 261–268
49. Morris P (1993) The breakout method for escaping from local minima, In Proceedings on 11th National Conference on Artificial Intelligence, Washington DC, pp. 40–45
50. Moscato P (1999) Memetic algorithms: A short introduction, In New Ideas in Optimisation, Corne et al. D (Eds.), pp. 219–234
51. Myung H and Kim JH (1996) Hybrid evolutionary programming for heavily constrained problems, Biosystems, 38(1), pp. 29–43
52. Neppalli VR, Chen CL, and Gupta JND (1996) Genetic algorithms for the two stage bicriteria flowshop problem, European Journal of Operational Research, 95, pp. 356–373
53. Von Neumann J (1966) In Burks A (Ed.), Theory of self reproducing automata, University of Illinois Press, Champaign, IL
54. Oman S and Cunningham P (2001) Using case retrieval to seedgenetic algorithms, International Journal of Computational Intelligence and Applications, 1(1), pp. 71–82
55. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Systems Magazine, 22(3), pp. 52–67
56. Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Stuttgart, Fromman-Holzboog
57. Schlottmann F and Seese D (2004) A hybrid heuristic approach to discrete multi-objective optimization of credit portfolios, Computational Statistics and Data Analysis, 47(2), pp. 373–399
58. Schwefel HP (1977) Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie, Basel, Birkhaeuser
59. Shi XH, Liang YC, Lee HP, Lu C, and Wang LM (2005) An improved GA and a novel PSO-GA-based hybrid algorithm, Information Processing Letters, 93(5), pp. 255–261
60. Sinha A and Goldberg DE (2003) A Survey of hybrid genetic and evolutionary algorithms, ILLIGAL Technical Report 2003004
61. Somasundaram P, Lakshmiramanan R, and Kuppusamy K (2005) Hybrid algorithm based on EP and LP for security constrained economic dispatch problem, Electric Power Systems Research, 76(1–3), pp. 77–85
62. Storn R and Price K (1997) Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, Journal of Global Optimization, 11(4), pp. 341–359
63. Swain AK and Morris AS (2000) A novel hybrid evolutionary programming method for function optimization, In Proceedings of the Congress on Evolutionary Computation (CEC2000), pp. 1369–1376
64. Tan KC, Yu Q, Heng CM, and Lee TH (2003) Evolutionary computing for knowledge discovery in medical diagnosis, Artificial Intelligence in Medicine, 27(2), pp. 129–154

65. Tseng LY and Liang SC (2005) A hybrid metaheuristic for the quadratic assignment problem, Computational Optimization and Applications, 34(1), pp. 85–113
66. Vargas LS, Quintana VH, and Vannelli A (1993) A tutorial description of an interior point method and its application to security-constrained economic dispatch, IEEE Transaction on Power Systems, 8(3), pp. 1315–1324
67. Vazquez M and Whitley D (2000) A hybrid genetic algorithm for the quadratic assignment problem, In Proceedings of Genetic and Evolutionary Computation Conference (GECCO), Morgan Kaufmann, San Mateo, CA, pp. 135–142
68. Zhong YW and Yang JG (2004) A hybrid genetic algorithm with Lamarckian individual learning for tasks scheduling, IEEE International Conference on Systems, Man and Cybernetics, Hague, Netherlands, pp. 3343–3348
69. Zmuda MA, Rizki MM, and Tamburino LA (2003) Hybrid evolutionary learning for synthesizing multi-class pattern recognition systems, Applied Soft Computing, 2(4), pp. 269–282
70. Yang M, Zhang X, Li X, and Wu X (2002) A hybrid genetic algorithm for the fitting of models to electrochemical impedence data, Journal of Electroanalytical Chemistry, 519, pp. 1–8
71. Wang L (2005) A hybrid genetic algorithm-neural network strategy for simulation optimization, Applied Mathematics and Computation, 170(2), pp. 1329–1343
72. Wang L, Zhang L, and Zheng DZ (2006) An effective hybrid genetic algorithm for flow shop scheduling with limited buffers, Computers and Operations Research, 33(10), pp. 2960–2971
73. Wolpert DH and Macready WG (1997) No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, 1(1), pp. 67–82
74. http://www.sciencedirect.com
75. http://www.springerlink.com
76. http://ieeexplore.ieee.org