Performance Evaluation and Population Reduction for a Self Adaptive Hybrid Genetic Algorithm (SAHGA)

Felipe P. Espinoza[1], Barbara S. Minsker[1], and David E. Goldberg[2]

[1] University of Illinois, Department of Civil and Environmental Engineering,
205 N. Matthews Ave, Urbana IL 61801
(fespinoz, minsker)@uiuc.edu

[2] University of Illinois, Department of General Engineering,
104 N. Matthews Ave, Urbana IL 61801
deg@uiuc.edu

Abstract. This paper examines the effects of local search on hybrid genetic algorithm performance and population sizing. It compares the performance of a self-adaptive hybrid genetic algorithm (SAHGA) to a non-adaptive hybrid genetic algorithm (NAHGA) and the simple genetic algorithm (SGA) on eight different test functions, including unimodal, multimodal and constrained optimization problems. The results show that the hybrid genetic algorithm substantially reduces required population sizes because of the reduction in population variance. The adaptive nature of the SAHGA algorithm together with the reduction in population size allow for faster solution of the test problems without sacrificing solution quality.

1 Introduction

Hybrid genetic algorithms (HGAs) are a natural extension to genetic algorithms (GA), combining the genetic algorithm's global search capabilities with the strengths of local search algorithms. These algorithms have been used in a number of different fields, including transportation engineering [8], water resources modeling [4], operations research [12], and groundwater management [11] to name a few. For example in the so-called "Bicriteria Transportation Problem" presented in [8], the GA was combined with the traditional simplex method to solve linear problems to create the HGA. Another example is the groundwater management problem. The HGA presented in [11] was created by combining the GA with constrained differential dynamic programming. In these applications and others, the HGA and the GA solved the problem using the same population size, and the local search step is applied in most of the cases to a small number of individuals in the population generation after generation. In this study, we analyze the effect of the local search component of the

HGA on the search and on reduction of the search space. This reduction reduces the population size necessary to solve the HGA problem relative to the SGA for the same solution reliability, thereby decreasing the computational burden.

Section 2 gives an overview of the HGA algorithms analyzed in this work and Section 3 presents the test functions that were used to evaluate performance of the HGA algorithms in Section 4. Finally, Section 5 presents our conclusions and recommendations.

2 SAHGA and NAHGA Algorithms

The two hybrid genetic algorithms (HGA) used in this work are NAGHA (non adaptive hybrid genetic algorithm) and SAGHA (self adaptive hydrid genetic algorithm) were originally proposed by [7]. A brief description of the algorithms is presented next. The full details of these algorithms can be found in [7].

2.1 Non-Adaptive Hybrid Genetic Algorithm (NAHGA)

The NAHGA algorithm combines a simple genetic algorithm (SGA) with local search. The local search step is defined by three basic parameters: local search frequency, probability of local search, and number of local search iterations. Local search frequency determines how frequently local search is invoked (e.g., every 3 iterations); probability of local search represents the fraction of individuals in the population that undergo local search at each invocation; and number of local search iterations represents the number of local search steps performed at each invocation.

2.2 Self-Adaptive Hybrid Genetic Algorithm (SAHGA)

The SAHGA algorithm works with the same operators as the NAHGA algorithm: local search frequency, probability of local search and number of local search iterations. The major difference in the approaches is that the SAHGA adapts in response to recent performance of the algorithm as it converges to the solution. In other words, the operators are used only when they can provide new information to the search. The adaptation process for the three parameters is different. The global-local switch is adapted by evaluating the ratio of the coefficient of variation of the fitness between two consecutives generations. Local search is performed when this ratio is greater than a specified local search threshold. This approach ensures that local search is only performed when the coefficient of variation is significantly increasing, which indicates that a new area of decision space is being searched and local search is needed. The second adaptive parameter is the probability of local search, which is decreased from the initial value at the beginning of every local search step. Finally, the number of local search steps is controlled by comparing the improvement attained in the local search step with the improvement attained by global search. When local search no longer improves average fitness more than the most recent global search iteration, the search reverts to global search.

3 Test Functions

In order to evaluate the performance of the algorithm, the following 8 test functions were selected from the GA literature:

- Unimodal problems: De Jong's 1 and De Jong's 2 [5]
- Multimodal problems with the same local mimimum at different locations: Branin [3] and Six Hump [6]
- 2 multimodal problems with different optima: Schwefel [17] and Griewank [1]
- 2 constrained unimodal problems: Test04 [13] and Bracken & McCormick [2]

4 Evaluation of HGA Performance

This section is divided in 5 sub-sections: local search algorithms, population size evaluation for the SGA, standard deviation reduction, population size for the HGA, and analysis of results.

4.1 Local Search Algorithm

The local search operator attempts to find the best solution starting at a previoulsy selected point, in this case a solution in the SGA population. For this analysis, 5 local search algortithms were selected. These algorithms are:

- Random Walk with Uniform Distribution (LS1): In general, the random walk is simply the movement from one point of the decision space to a new point randomly selected using a uniform distribution from a neighborhood around the starting point [18]. One iteration of this algorithm requires one fitness function evaluation.
- Random Walk with Normal Distribution (LS2): This algorithm is similar to the uniform distribution discussed previously, but the change of location is evaluated with a normal distribution instead of a uniform distribution. For this reason, the points located near the starting point are more likely to be selected than those located closer to the boundary of the search area. Again, one iteration requires one function evaluation.
- (1+1)-Evolutionary Strategy (LS3): This algorithm, proposed by [14] and [16], randomly selects a new location using a normal distribution with variable standard deviation. The standard deviation changes following the so-called $1/5$ success rule based on the evaluation of success of the search. This algorithm also requires one fitness function evaluation per local search iteration.
- Random Derivative (LS4): This algorithm randomly selects a search direction, and using this direction, the location of a new point is evaluated. This algorithm needs 2 fitness function evaluation for every local search iteration (one for the evaluation of the coordinates of the new point and one for its fitness).
- Steepest Descent (LS5): The algorithm evaluates the new point following the direction of the gradient of the function at the starting point. This algorithm performs one function evaluation for every one of the decision variables of the

particular test problem (to numerically evaluate gradient) and one function evaluation to evaluate the fitness of the new individual.

4.2 Population Size for Genetic Algorithm

One of the critical elements for optimal performance of a GA is the population size. For the population size evaluation, a relationship derived from the random walk model proposed by [9] is used. In their approach the population size is given by:

$$N \geq -2^{K-1}\, ln(\alpha)\left(\frac{\sigma_f}{d}\right) \quad (1)$$

In Eq. (1), K represents the building block (BB) order, α is the reliability (i.e., the probability that the GA finds the optimal solution); σ_f is the standard deviation of the fitness function, and d is the signal difference between the best and second best solution. The parameters σ_f and d are estimated using a large, random initial population. For these test problems, d was estimated from the differences in fitness of the best members of the population. In this case the analysis was performed by means of a probabilistic approach based on frequency analysis theory [10], in which the histogram of the fitness function is evaluated. This histogram represents different "classes" of fitness that can be statistically identified. Using the histogram, "d" is evaluated as the size of the first class of the histogram. The building block order, K, is unknown but can be assumed to vary between 1 and 5 [15].

Using this information, a range of initial population sizes for the SGA was estimated from Eq. (1) for all 8 test problems for differrent values of parameter K, as shown in Table 1.

The final step in the analysis is the selection of the starting population from the values given in Table 1. For this analysis, the convergence time must be less than the drift time. The convergence time is evaluated with the relation $t \approx 2l$ (where l is the chromosome length) [19]. The drift time is given by $t_{drift} \approx 1.4\ N$ (where N is the population size) [20]. Imposing the condition that convergence time must be less than drift time to obtain the correct solution, the population size must satisfy the relation presented in Eq. (2) (lower boundary for population size):

$$N > 1.42\, l \quad (2)$$

Table 1: Population size (SGA) for different values of K

Building Block Order (K)	Test Function							
	DJ1	DJ2	Branin	Six-Hump	Schwefel	Griewank	Test04	B&M
1	20	12	20	10	35	20	16	24
2	40	24	40	20	70	40	32	48
3	80	48	80	40	140	80	64	96
4	160	96	160	80	280	160	128	192
5	320	192	320	160	560	320	256	384

Table 2 shows the chromosome length for each test function and the corresponding lower boundary for the population evaluated with Eq. (2). The table

also includes the minimum building block order that satisfies Eq. (2) (see Table 1) and the adopted population for the SGA runs.

Table 2: Adopted SGA Population for all 8 problems

Function	Chromosome Length	Lower Boundary	Minimum Building Block Order (K)	Adopted Population
DJ1	150	213	5	320
DJ2	60	83	4	96
Branin	60	83	4	160
Six-Hump	60	83	5	160
Schwefel	150	213	4	280
Griewank	60	86	4	160
Test04	60	83	4	128
B&M	60	86	3	96

4.3 Standard Deviation Reduction: Local Search Effect on Population Size

For the HGA case, Eq. (1) is modified to include the effect of local search. During local search, the diversity of the initial population is decreased because multiple members of the population usually have the same nearby local optima. The following example contains the results of local search on the multimodal function Griewank for a population of 20 individuals. Fig. 1 shows the fitness for a random population before and after local search (uniform random and gradient). The line represents the average population fitness. From Fig. 1, it is clear that the standard deviation of the fitness for the population after local search is substantially reduced. Initially the average is 30 and the standard deviation is 27.1. After 3 iterations (60 function evaluations) uniform random search reduces the standard deviation to 24.6 and the average to 25.2 (see open squares and dashed line). Steepest descent local search performed 1 iteration (60 function evaluations) and changed the standard deviation to 7.3 and reduced the average to 12.1 (see open circles and solid line). Steepest descent local search lowered the standard deviation and the average by an amount larger than uniform random local search while using the same 60 function evaluations. Both methods of local search performed the same number of function evaluations, however the gradient method only did 1 iteration while the uniform random method did 3 iterations in order to evaluate the results for the same amount of effort. These results show that gradient search decreases diversity faster than random search.

The local search reduction effect can be modeled as $\sigma_{fl} = \beta\, \sigma_f$, where $0 \le \beta < 1$ is the standard deviation reduction by local search. Eq. (1) can be rewritten as:

$$N \ge -2^{K-1}\, ln(\alpha) \left(\beta \frac{\sigma_f}{d}\right) \tag{3}$$

Eq. (3) shows that the population for the HGA algorithms should be directly proportional to the SGA population.

The parameter β can be estimated by applying local search to the initial random population for a predefined number of iterations. For the current application, local

search was applied to a population of 1,000 individuals for a total of 10 iterations. Fig. 2 shows the change of standard deviation with respect to the standard deviation of the initial population for the selected test problem (Griewank). Since most of the local search steps involved fewer than 6 iterations, the value used for posterior evaluations was the average value of β across the first 5 iterations.

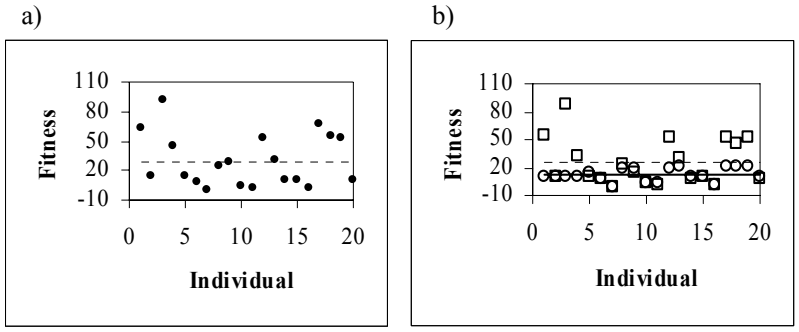

Fig. 1. Local search effect on fitness: a) before local search, b) after local search

In Fig. 2, gradient (LS5) has the largest effect on standard deviation;1 iteration of this algorithm is equivalent to 3 iterations of LS1, LS2 or LS3. The random derivative (LS4) method is the second most effective local search method at decreasing the standard deviation of the population. The first three local search methods also have substantial effects on the standard deviation, especially if the amount of effort involved is taken into consideration.

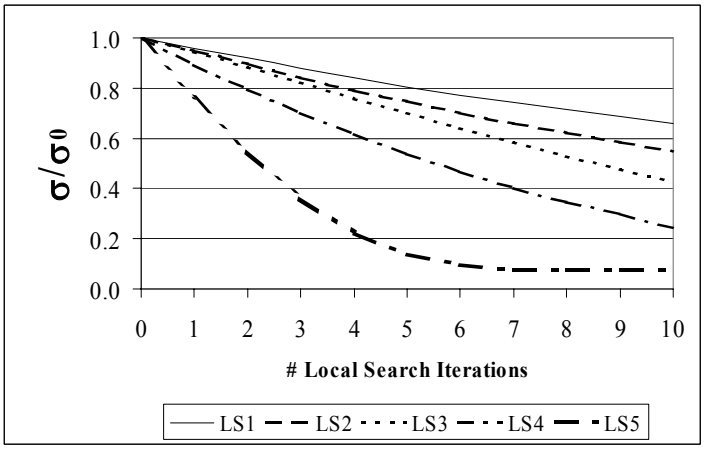

Fig. 2: Standard Deviation Reduction, β, for Griewank Function

4.4 Population Size for Hybrid Algorithms

Using Eq. (3) and the value of β estimated for every one of the 5 algorithms, Table 3 displays the HGA population results for every test problem and local search algorithm. From the results, it is clear that the effect of the different local search algorithms is the same for DJ1 and Six-Hump. On the other hand, for Griewank the population size is different for every one of the 5 local search algorithms in consideration because of the different reductions in standard deviation.

Table 3: HGA Population

	Local Search Algorithm				
Test Function	**LS1**	**LS2**	**LS3**	**LS4**	**LS5**
DJ1	288	288	288	288	288
DJ2	64	64	64	40	40
Branin	96	96	80	48	48
Six-Hump	96	96	96	96	96
Schwefel	240	224	216	160	160
Griewank	144	136	128	112	64
Test04	120	112	112	112	80
B&M	92	88	88	48	48

4.5 HGA Performance

The HGA performance results consist of 2 types of data: results evaluated for one population for a default set of parameters defined in [7], and results evaluated using Monte Carlo simulation for different combinations of parameters also defined in [7].

The first set of results for Griewank can be presented in two graphs. The first graph (Fig. 3) plots the best fitness in the population during the SGA and SAHGA search processes, with SAHGA using the 5 local search techniques. The results are presented in terms of number of function evaluations because the adaptive local search uses different numbers of function evaluations in each generation. The results presented also only include the results attained by the application of SAHGA because SAHGA performance is consistently better for the 8 test problems.

Fig. 3 shows that the SGA is inferior to SAHGA with all five local search algorithms. SGA required the most function evaluations with a total of 8,800. Steepest descent (LS5) performed only 4,896 function evaluations with a population of 64, a reduction of 44%. The steepest descent method's speed and its ability to reduce the required population offset the fact that it is the most expensive method. In fact, its performance is better than the performance of normal random search (LS2) and (1+1)-ES (LS3).

The second result, shown in Fig. 4, is the convergence ratio, which is defined as the number of individuals with fitness equal to or near the fitness of the best individual. Fig. 4 shows that the results attained by SAHGA are again better than the SGA for all 5 local search approaches. The results from the other 7 test functions are

not shown because they present the same trend, with SAHGA performing better than the SGA by 5% to 33%.

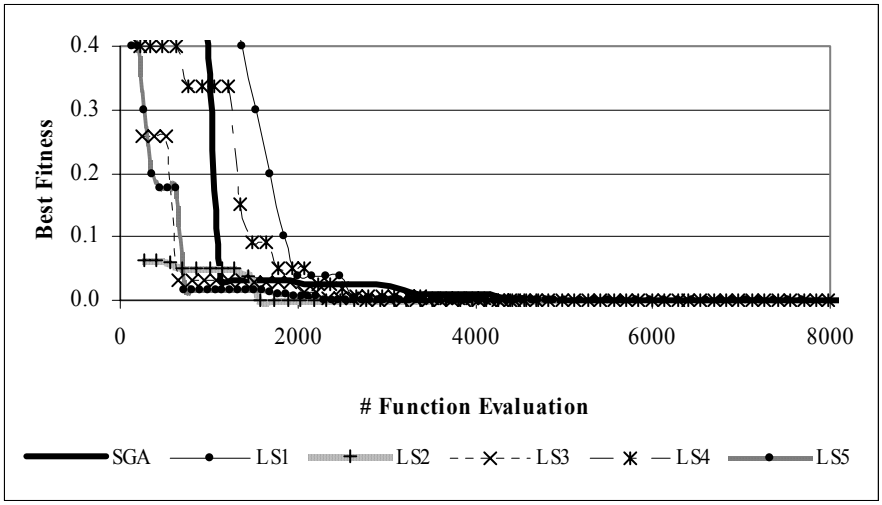

Fig. 3. Best Fitness for Griewank

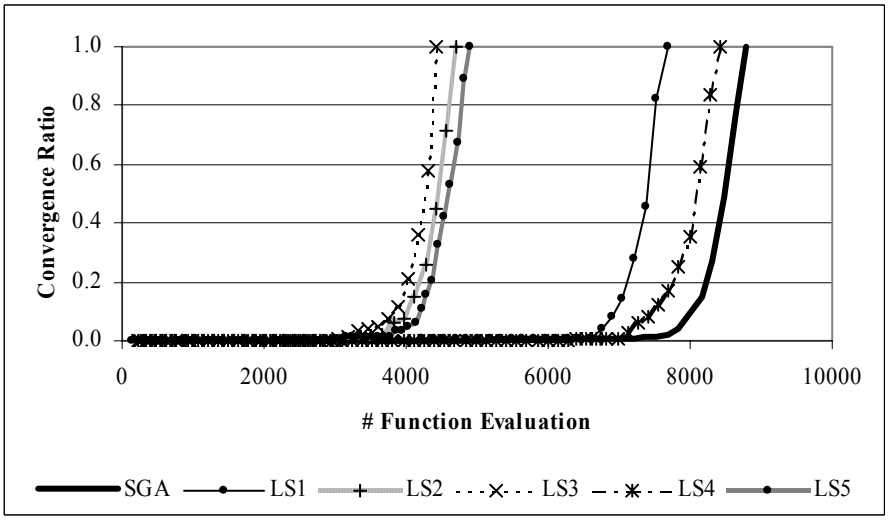

Fig. 4. Convergence Ratio for Griewank

The solution quality of the algorithms was also evaluated by performing 100 Monte Carlo simulations over the initial population for every one of the parameter combinations described in [7]. Table 4 shows the results of these simulations. The results are presented for the best, default and worst parameter combinations. For

SAHGA, the average number of function evaluations for the default and best set of parameters are always better than the results attained with the SGA, and the difference between the best set of results and the default varies between 2% and 8%. On the other hand, for NAHGA sometimes the results associated with the default set of parameters are worse than the results attained with the SGA, and the difference between the best set of results and the default set of results varies between 10% and 25% for different local search algorithms. This shows that parameter estimation is critical to attaining good performance for the NAHGA, but the adaptive capabilities of SAHGA reduce the need for careful parameter selection. For both algorithms, the worst set of results was attained when all the individuals are undergoing local search.

Table 4: Monte Carlo Simulation Results
Average Number of Function Evaluations

	SGA	SAHGA			NAHGA		
Local Search		**Worst**	**Default**	**Best**	**Worst**	**Default**	**Best**
LS1	4,326	6,851	4,020	3,906	9,462	4,373	3,936
LS2	4,326	6,470	3,800	3,720	8,936	4,131	3,717
LS3	4,326	6,402	3,775	3,660	8,723	4,036	3,478
LS4	4,326	13,776	3,964	3,666	14,337	4,645	3,508
LS5	4,326	7,905	3,503	3,292	8,961	4,153	3,071

For the other test problems in this study, the results are similar. The only difference is that the best local search algorithm sometimes varies among gradient, (1+1)-ES, uniform random walk, normal random walk, and random derivative. This phenomenon is related to the differences in standard deviation reduction of each algorithm for different test functions. For example, when the differences in standard deviation reduction between steepest descent and random search is not significant (see Table 3 for functions DJ1 and Six-Hump), random search is more likely to perform better than steepest descent given the big difference in the effort necessary to apply the algorithms. On the other hand, when the difference is significant (see Griewank function in Table 3), steepest descent is better than random walk.

Fig. 5 shows the reliability of SGA, SAHGA and NAHGA for the default set of parameters. SAGHA reliability was in the range of 98-100%, while solving the problem over a much smaller population than the SGA, which means it is less expensive. NAGHA is very efficient as well, with reliability values of 96% to 100%, again with smaller population sizes than the SGA. SAHGA and NAHGA have somewhat lower reliabilities for LS4 and LS5 because the extensive standard deviation reduction associated with the gradient-based algorithms sometimes causes premature convergence due to loss of diversity in the population.

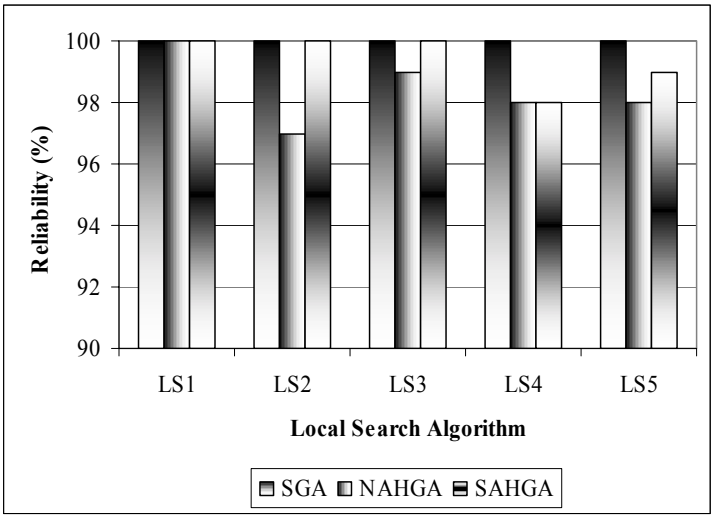

Fig. 5. SGA and HGA Reliabilities for Default Set of Parameters

5 Recommendations and Conclusions

The results presented in this paper indicate that the local search capabilities of the HGA algorithm enabled robust solution of complex, unimodal, multi-modal, and constrained problems with less effort than the SGA. Of the two hybrid algorithms, SAHGA is always superior to NAHGA because near-optimal performance is attained for a broad range of parameters. Another result is related to the importance of the local search algorithm selected. The results show that the best performance is not always attained for the same local search algorithm across all test functions, mostly because of the difference in effort necessary to apply the different local search algorithms. For any function, the most suitable local search algorithm can be pre-selected using only the population sizing analysis shown in Table 3. When the difference between populations (for different local search algorithms) is not significant, it is more likely that a random search algorithm (which requires only 1 function evaluation per local search iteration) will perform better than steepest descent (which requires n+1 function evaluations per iteration, with n the number of decision variables).

Another important result derived from this research is related to the setting of population sizing in Section 4. This process allows us to evaluate a good estimate of the population to achieve optimal performance, but there is a possibility that the same level of reliability is achieved with a smaller population. With respect to the HGA, the analysis presented shows clearly how local search reduces the standard deviation of fitness in the population, which in turn reduces the required population size to achieve the same level of reliability. This reduced population together with the HGA reduces

the total number of function evaluations by 44% on average. In this way, the HGA is almost twice as fast as the original GA.

The next step of this research will be to analyze possible ways to improve the performance of the algorithm with a more detailed study of the processes involved. Finally, the algorithm will be applied to solve real world problems.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. BES 97-34076 CAR and the U. S. Army Research Office under Grant No. DAAD 19-00-7-0389.

Disclaimer

References

1. Bäck , T., D. Fogel and Z. Michalewicz, (eds): Handbook of Evolutionary Computation, Bristol and New York. Institute of Physics Publishing Ltd and Oxford University Press (1997)

2. Bracken, J. G. P. McCormick: Ausgewählte Nwendungen Nichtlinearer Programmierung. Berliner Union and Kohlhammer, Sttutgart (1970)

3. Branin, F. K.: A Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations. IBM J. Res. Develop., pp. 504-522. (1972)

4. Cai, X., McKinney, D. and Lasdon. L.: Solving Nonlinear Water Management Models Using a Combined Genetic Algorithm and Linear Programming Approach. Advances in Water Resources, 24, 667-676. (2001)

5. De Jong, K. A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Dissertation, University of Michigan, Ann Arbor, MI. (1975)

6. Dixon, L. C. W. and Szego, G. P.: The Optimization Problem: An Introduction. In Dixon, L. C. W. and Szego, G. P. (Eds.): Towards Global Optimization II, New York: North Holland. (1978)

7. Espinoza, F., B. S. Minsker, and D. Goldberg. (2001). “A Self Adaptive Hybrid Genetic Algorithm”. L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. 2001. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2001. San Francisco, Morgan Kaufmann Publishers.

8. Gen, M., Ida, K., and Li, Y.: Bicriteria Transportation Problem by Hybrid Genetic Algorithm. Computers & Industrial Engineering, 35(1-2), 363-366. (1998)

9. Harik, G.R., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L.: The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations.” In Proceedings of the 1997 IEEE Conference on Evolutionary Computation, pp. 7-12, IEEE Press, New York, NY. (1997)

10. Hogg, R., and Craig: A. Introduction to Mathematical Statistics. Macmillan Publishing Co., Inc., New York. (1978)

11. Hsiao, C. and Chang, L.: Dynamic Optical Groundwater Management With Inclusion Of Fixed Costs. Journal of Water Resources Planning and Management, ASCE, 128(1), 57-65. (2002)

12. Lin, W., Delgado-Frias, J, Gause, D., and Vassiliadis, S.: Hybrid Newton-Raphson Genetic Algorithm for the Traveling Salesman Problem. Cybernetics & Systems, 26(4), 387-412. (1995)

13. Kim, J-H. and H. Myung: Evolutionary Programming Techniques for Constrained Optimization Problems. Evolutionary Computation, 1(2), 129-140. (1997)

14. Rechenberg, I.: Ecolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologishen Evolution. Frommann-Iolzboog Verlag, Stuttgart. (1973)

15. Reed, P., Minsker, B. S., and Goldberg, D. E.: Designing a Competent Simple Genetic Algorithm for Search and Optimization. Water Resources Research, 36(12), 3757-3761. (2000)

16. Schwefel, H. P.: Evolutionsstrategie und Numerische Optimierung. PhD dissertation, Department of Process Engineering, Technical University of Berin, Berlin, Germany. (1975)

17. Schwefel, H. P.: Numerical Optimization of Computer Models. John Wiley & Sons, Chichester-New York-Brisbane-Toronto, (1981)

18. Spitzer, F.: Principles of random walk. D. Van Nostrand Company, Inc. (1964)

19. Thierens, D., Goldberg, D. E., and Guimaraes Pereira: A. Domino Convergence, Drift, and the Temporal-Salience Structure of Problems. In The 1998 IEEE International Conference on Evolutionary Computation Proceedings, pp. 535-540, IEEE Press, New York, NY, (1998)