A Look-Ahead Based Meta-Heuristics for Optimizing Continuous Optimization Problems

Nordli, Thomas; Bouhmala, Noureddine
Department of Microsystems - University of South-Eastern Norway

Nordli, T., & Bouhmala, N. (2022). A Look-Ahead Based Meta-Heuristics for Optimizing Continuous Optimization Problems. In *International Conference on Optimization, Learning Algorithms and Applications* (pp. 48-55). Springer, Cham. https://doi.org/10.1007/978-3-030-91885-9_4

A Look-Ahead Based Meta-Heuristics for Optimizing Continuous Optimization Problems

Thomas Nordli and Noureddine Bouhmala

University of South-Eastern Norway, Norway
http://www.usn.no

Abstract. In this paper, the famous kernighan-Lin algorithm is adjusted and embedded into the simulated annealing algorithm and the genetic algorithm for continuous optimization problems. The performance of the different algorithms are evaluated using a set of well known optimization test functions.

Keywords: Continuous optimization problems · Simulated annealing · Genetic algorithm · Local search

1 Introduction

Several types of meta-heuristics methods have been designed for solving continuous optimization problems. Examples include genetic algorithms [8][9] artificial immune systems [7], and taboo search[5]. Meta-heuristics can be divided into two different classes. The first class refers to single-solution search algorithms. A notable example that belongs to this class is the popular simulated annealing algorithm (SA) [12], which is a random search that avoids getting stuck in local minima. In addition to solutions corresponding to an improvement in objective function value, SA also accepts those corresponding to a worse objective function value using a probabilistic acceptance strategy.

The second class of algorithms refer to population based algorithms. Algorithms of this class applies the principle of *survival of the fittest* to a population of potential solutions, iteratively improving the population. During each generation, pairs of solutions called individuals are generated to breed a new generation using operators borrowed from natural genetics. This process is repeated until a stopping criteria has been reached. Genetic algorithm is one among many that belongs to this class. The following papers [2][1][6] provide a review of the literature covering the use of evolutionary algorithms for solving continuous optimization problems. In spite of the advantages that meta-heuristics offer, they still suffer from the phenomenon of premature convergence.

Recently, several studies combined meta-heuristics with local search methods, resulting in more efficient methods with relatively faster convergence, compared to pure meta-heuristics. Such hybrid approaches offer a balance between diversification — to cover more regions of a search space, and intensification – to find better solutions within those regions. The reader might refer to [3][14] for further reading on hybrid optimization methods.

This paper introduces a hybridization of genetic algorithm and simulated annealing with the variable depth search (VDS) Kernighan-Lin algorithm (KL) (which was firstly presented for graph partitioning problem in [11]). Compared to simple local search methods, KL allows making steps that worsens the quality of the solution on a short term, as long as the result gives an improvement in a longer run.

In this work, *the search for one favorable move in SA*, and *one two-point cross-over in GA*, are replaced by a *search for a favorable sequence of moves in SA* and *a series of two-point crossovers* using the objective function to guide the search.

The rest of this paper is organized as follows. Section 2 describes the combined simulated annealing and KL heuristic, while Section 3 explains the hybridization of KL with the genetic algorithm. Section 4 lists the functions used in the benchmark while Section 5 shows the experimental results. Finally, Section 6 concludes the paper with some future work.

2 Combining Simulated Annealing with Local Search

Previously, SA in combination with KL, was applied for the Max-SAT problem in [4].

SA iteratively improves a solution by making random perturbations (moves) to the current solution — exploring the neighborhood in the space of possible solutions.

It uses a parameter called *temperature* to control the decision whether to accept bad moves or not. A bad move is a solution that decreases the value of the objective function.

The algorithm starts of with a high temperature, when that almost all moves are accepted. For each iteration, the temperature decreases, the algorithm becomes selective, giving higher preference for better solutions.

Assuming an objective function f is to be maximized. The algorithm starts computing the initial temperature T, using a procedure similar to the one described in [12]. The temperature is computed such that the probability of accepting a bad move is approximately equal to a given probability of acceptance Pr. First, a low value of is chosen as the initial temperature. This temperature is used during a number of moves.

If the *ratio of accepted bad moves* is less than Pr, the temperature is multiplied by two. This continues until the observed acceptance ratio exceeds Pr. A random starting solution is generated and its value is calculated. An iteration of the algorithm starts by performing a series of so-called KL perturbations or moves to the solution S_{old} leading to a new solution S^i_{new} where i denotes the number of consecutive moves. The change in the objective function called *gain* is computed for each move. The goal of KL perturbation is to generate a sequence of objective function scores together with their corresponding moves. KL is supposed to reach convergence if the function scores of five consecutive moves are bad moves. The subset of moves having the best cumulative score $BCS^k_{(SA+KL)}$

is identified. The identification of this subset is equivalent to choosing k so that $BCS^k_{(SA+KL)}$ in equation 1 is maximum,

$$BCS^k_{(SA+KL)} = \sum_{i=1}^{k} gain(S^i_{new}) \tag{1}$$

where i represents the i^{th} move performed, k the number of a moves, and $gain(S^i_{new}) = f(S^i_{new}) - f(S^{i-1}_{new})$ denotes the resultant change of the objective function when i^{th} move has been performed. If $BCS^k_{(SA+KL)} > 0$, the solution is updated by taking all the perturbations up to the index k and the best solution is always recorded. If $(BCS^k_{SA+KL} \leq 0)$, the simulated acceptance test is restricted to only the resultant change of the first perturbation. A number from the interval (0,1) is drawn by a random number generator. The move is accepted ff the drawn number is less than $exp^{-\delta f/T}$. The process of proposing a series of perturbations and selecting the best subset of moves is repeated for a number of iterations before the temperature is updated. A temperature reduction function is used to lower the temperature. The updating of the temperature is done using a geometric cooling, as shown in equation 2

$$T_{new} = \alpha \times T_{old} \tag{2}$$

, where $\alpha = 0.9$.

3 Combining Genetic Algorithm with Local Search

Genetic Algorithm (GA) belong to the group of evolutionary algorithms. It works on a set of solutions called a population. Each of these members, called chromosomes or individuals, is given a score (fitness), allowing the assessing of its quality. The individuals of the initial population are in most cases generated randomly. A reproduction operator selects individuals as parents, and generates off-springs by combining information from the parent chromosomes. The new population might be subject to a mutation operator introducing diversity to the population. A selection scheme is then used to update the population — resulting in a new generation. This is repeated until the convergence is reached — giving an optimal or near optimal solution.

The simple GA as described in **??** is used here. It starts by generating an initial population represented by floating point numbers. Solutions are temporary converted to integers when bit manipulation is needed, and resulting integers are converted back to floating point representation for storage. A roulette function is used for selections. The implementation and based on the one described in section IV of [10], where more details can be found.

The purpose of KL-Crossover is to perform the crossover operator a number of times generating a sequence of fitness function scores together with their corresponding crossover. Thereafter, the subset of consecutive crossovers having the best cumulative score $BCS^k_{(GA-KL)}$ is determined. The identification of this subset is the same as described in SA-KL. GA-KL chooses k so that $BCS^k_{(GA+KL)}$ in

equation 4 is maximum, where CR^i represents the i^{th} crossover performed on two individuals, I_l and I_m, k the number of allowed crossovers, and $gain(I_l, I_m)_{CR^i}$ denotes the resulting change in the fitness function when the i^{th} crossover CR^i has been performed calculated shown in equation 3.

$$gain(I_l, I_m)_{CR^i} = f(I_l, I_m)_{CR^i} - f(I_l, I_m)_{CR^{i-1}}, \tag{3}$$

where CR^0 refers to the chosen pair of parents before applying the cross-over operator. KL-crossover is supposed to reach convergence if the gain of five consecutive cross-overs is negative. Finally, the individuals that are best fit are allowed to move to the next generation while the other half is removed and a new round is performed.

$$BCS^k_{(GA+KL)} = Max\left[\sum_{i=1}^{k} gain(Individual_l, Individual_m)_{CR^i}\right]. \tag{4}$$

4 Benchmark Functions & Parameter Setting

The stopping criterion for all four algorithms (SA, SA-Look-Ahead,, GA, GA-Look-Ahead) is supposed to be reached if the best solution has not been improved during 100 consecutive iterations for (SA, SA-Look-Ahead) or 10 generations for (GA,GA-Look-Ahead). The starting temperature for (SA,Look-Ahead-SA) is set to 0.8 (i.e., a bad move has a probability of 80% for being accepted). In the inner loop of (SA,Look-Ahead-SA), the equilibrium is reached if the number of accepted moves is less than 10%.

The ten following benchmark functions were retrieved from [13] and tested.

1: Drop Wave	$f(x,y) = -\frac{1+\cos(12\sqrt{x^2+y^2})}{\frac{1}{2}(x^2+y^2)+2}$
2: Griewangk	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
3: Levy Function	$\sin^2(3\pi x) + (x-1)^2[1+\sin^2(3\pi y)] + (y-1)^2[1+\sin^2(2\pi y)]$
4: Rastrigin	$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos 2\pi x_i)$
5: Sphere Function	$f(x) = \sum_{i=1}^{n} x_i^2$
6: Weighted Sphere Function	$f(x,y) = x^2 + 2y^2$
7: Sum of different power functions	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$
8: Rotated hyper-ellipsoid	$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{i} x_j^2$
9: Rosenbrock's valley	$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1-x_i)^2]$
10:Three Hump Camel Function	$f(x,y) = x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$

5 Experimental Results

The results are visualized in figure 1, 2 and 3, where both the the *mean* and the *best* solution *over 100 runs* are plotted. The X-axis shows the number of generations (for GA and GA-KL — figure 1 and 2) or the iterations (for SA and

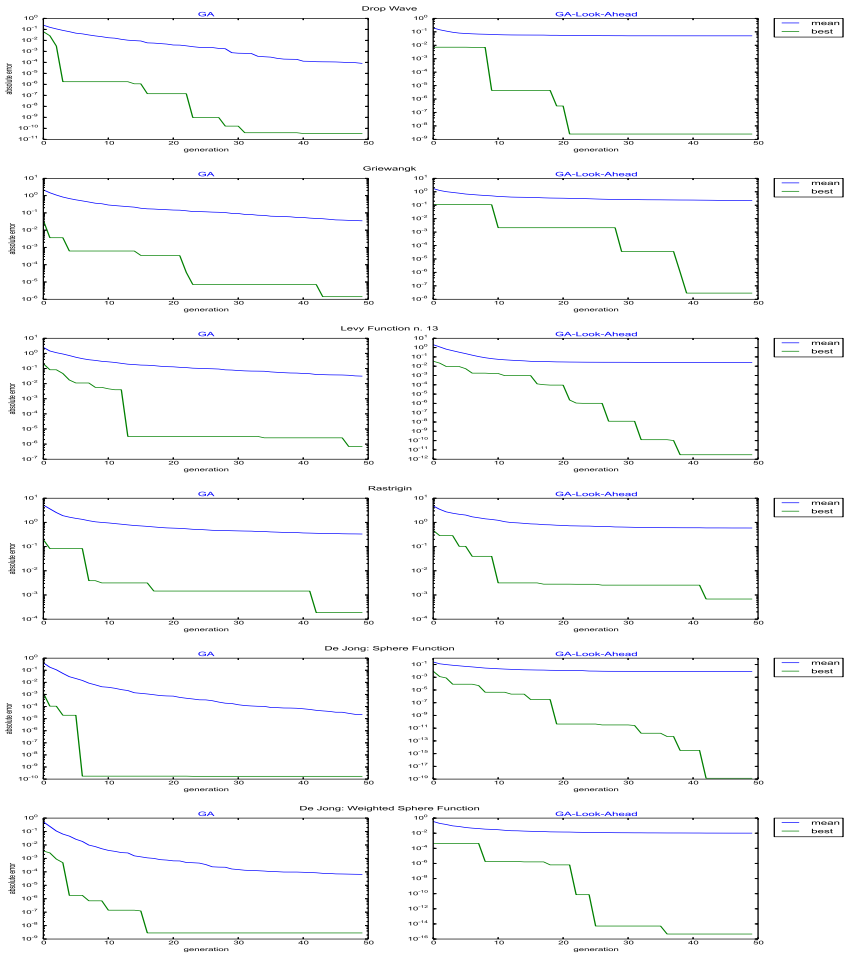

Fig. 1. Functions 1–6: GA vs. GA-KL

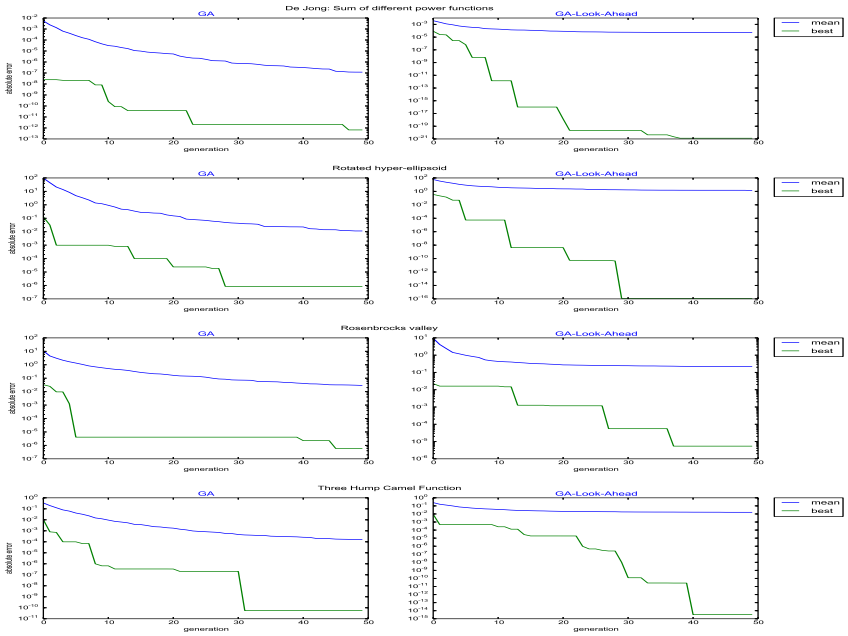

Fig. 2. Functions 7–10: GA vs. GA-KL

SA-KL — figure 3), while the Y-axis gives the absolute error (i.e. the excess of deviation from the optimal solution).

Figures 1 and 2 compares GA against GA-KL (GA-Look-Ahead). Looking at the mean solution, GA delivers on average lower absolute error solutions on almost 9 cases out of 10. The average percentage change error reduction in favor of GA is 5% for function 1, 12% for function 2, 43% for function 4, within 1% for functions 5, 6 , 7 and 8, 14% for function 9 and finally 2% for 10. Function 3 was the only test case where GA-Look-Ahead wins with an average percentage change error reduction of 26%. On the other hand, comparing the curves representing the evolution of the (best solution) fittest individual produced by the two algorithms, GA-Look-Ahead is capable of reaching solutions of higher precision when compared to GA (10^{-8} versus 10^{-6} for function 2, 10^{-11} versus 10^{-6} for function 3, 10^{-19} versus 10^{-10} for function 5, 10^{-15} versus 10^{-9} for function 6, 10^{-21} versus 10^{-12} for function 7, 10^{-16} versus 10^{-6} for function 8, 10^{-14} versus 10^{-10} for function 10. The diversity of the population produced by GA-Look-Ahead enables GA from premature convergence phenomenon leading GA to continue for more generations before convergence is reached. GA-Look-Ahead performed 69% more generations compared to GA for function 3, 82% for function 5, 57% for function 6, and 25% for function 10.

Figure 3 shows the results for SA and SA-KL (SA-Look-Ahead). Looking at the evolution of the mean, both methods produce similar quality of results while SA is showing insignificantly lower absolute error on few cases. However, when

comparing the best versus the best, SA-Look-Ahead delivers solutions of better precision than SA. The precision rate is 10^{-10} versus 10^{-8} for function 1, 10^{-11} versus 10^{-6} for function 4, 10^{-15} versus 10^{-9} for function 7, and 10^{-7} versus 10^{-4} for function 9.

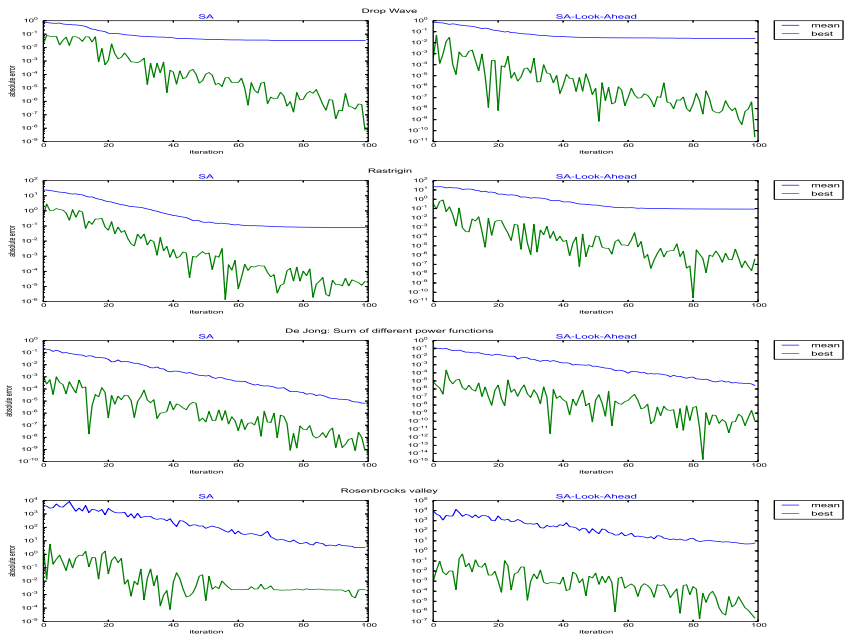

Fig. 3. Functions 1, 4, 7, 9: SA vs. SA-KL

6 Conclusion

Meta-heuristics global optimization algorithms have become widely popular for solving global optimization problems. In this paper, both SA and GA have been combined for the first time with the popular Kernighan-Lin heuristic used for the graph partitioning problem. The main idea is to replace the search for one possible perturbation in SA or one typical crossover in GA by a search for favorable sequence of perturbations or crossovers using the objective function to be optimized to guide the search. The results presented in this paper show that the proposed scheme enables both SA and GA to reach solutions of higher accuracy which is the significant result of the present study. In addition, the proposed scheme enables GA to maintain the diversity of the population for a longer period preventing GA from reaching premature convergence which still remains a serious defect in GA when applied to both continuous and discrete optimization. With regard to the future, we believe that the performance of the strategy could

be further improved by selecting the type of perturbations to be processed by introducing a probability of acceptance at the level of the Kernighan-Lin heuristic whenever a bad move is to be selected.

References

1. Ardia, D., Boudt, K., Carl, P., Mullen, K., Peterson, B.G.: Differential evolution with deoptim: an application to non-convex portfolio optimization. The R Journal **3**(1), 27–34 (2011)
2. Ardia, D., David, J., Arango, O., Gómez, N.D.G.: Jump-diffusion calibration using differential evolution. Wilmott **2011**(55), 76–79 (2011)
3. Arun, N., Ravi, V.: Aconm: A hybrid of ant colony optimization and nelder-mead simplex search. Institute for Development and Research in Banking Technology (IDRBT), India (2009)
4. Bouhmala, N.: Combining simulated annealing with local search heuristic for max-sat. Journal of Heuristics **25**(1), 47–69 (2019)
5. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. European journal of operational research **123**(2), 256–270 (2000)
6. Chelouah, R., Siarry, P.: Genetic and nelder–mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. European Journal of Operational Research **148**(2), 335–348 (2003)
7. De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE transactions on evolutionary computation **6**(3), 239–251 (2002)
8. Goldberg, D.E.: Genetic algorithms in search. Optimization, and Machine Learning (1989)
9. Holland John, H.: Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press (1975)
10. Jensen, B., Bouhmala, N., Nordli, T.: A novel tangent based framework for optimizing continuous functions. Journal of Emerging Trends in Computing and Information Sciences **4**(2) (2013)
11. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell system technical journal **49**(2), 291–307 (1970)
12. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
13. Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: Test functions and datasets. Retrieved from http://www.sfu.ca/ ssurjano
14. Tank, M.: An ant colony optimization and nelder-mead simplex search hybrid algorithm for unconstrained optimization (2009)