Using Polynomial Local Search and Kolmogorov Complexities to Better Understand Evolutionary Algorithms

Natalio Krasnogor
Automatic Scheduling, Optimisation and Planning Group
School of Computer Science and IT
University of Nottingham, U.K.
http://slater.chem.nott.ac.uk/~natk/Public/index.html
natalio.krasnogor@nottingham.ac.uk

Abstract

In [5] we develop a syntax-only classification of evolutionary algorithms, particularly, the so called Memetic Algorithms(MAs)[11]. When "syntactic sugar" is added to our model, we are able to investigate the Polynomial Local Search (PLS) complexity of Memetic Algorithms. In this paper we enunciate the PLS-Completeness of not just one PLS problem but of whole classes of problems associated with MAs applied to the TSP. These classes of problems arise when a range of mutation, crossover and local search operators are considered. Our PLS-Completeness results shed light on the worst case behavior that can be expected of an MA that belongs to the classes of algorithms analyzed. We also mention some intriguing connections between the Polynomial Local Search Complexity theory and Kolmogorov Complexity theory in the context of evolutionary algorithms understanding.

Memetic_Algorithm():
Begin
/* P is a (multi)set of individuals (i.e. solutions) */
$t = 0$;
/* We put the evolutionary clock to zero */
Randomly generate an initial population $P(t)$;
Repeat Until (Termination Criterion Fulfilled) **Do**
Compute the fitness $f(p)$ $\forall p \in P(t)$;
Choose a subset of $P(T)$ (biased by $f(p)$), put in $M(t)$;
Recombine and variate individuals in $M(t)$, put in $M'(t)$;
Improve_by_local_search($M'(t)$);
Compute the fitness $f(p)$ $\forall p \in M'(t)$;
Generate $P(t+1)$ from individuals in $P(t)$ and $M'(t)$;
$t = t + 1$;
Od
Return best $p \in P(t-1)$;
End.

Figure 1: A basic version of a memetic algorithm.

1 Memetic Algorithms

Memetic Algorithms are metaheuristics designed to find solutions to complex and difficult optimization problems. They are evolutionary algorithms that include a stage of individual optimization or learning as part of their search strategy. A simple Memetic Algorithm scheme is shown in Fig. 1. As almost all evolutionary algorithms[1] an MA keeps a (multi)set of solutions, called the population, and applies to it several perturbation operators, e.g., mutation, crossover and (in the Memetic Algorithms case) local search. The perturbed solutions are then compared to the original ones and the "fittest" solutions are chosen for a further round of perturbations. The rationale underlying this is that by presenting solutions to selection pressure those which are better than the original ones will survive and will keep improving. The inclusion of a local search stage into the traditional evolutionary cycle of crossover-mutation-selection is not a naive or a minor change of the evolutionary algorithm architecture, on the contrary, it is a crucial deviation that affects how local and global search is performed[3],[7],[10],[5]. The reader should also note that the pseudocode shown in Fig. 1 is just one possible way to hybridize a genetic algorithm with local search. In fact, a great number of distinct memetic algorithms' architectures has been presented in the literature and integrated into a formal model [5].

2 PLS and Kolmogorov Concepts In the Understanding of Memetic Algorithms Behavior

We assume here that the reader is familiar with the basic concepts of Polynomial Local Search Complexity theory [4][17] and Kolmogorov complexity theory [9]. Please note that there is a large literature body on these topics.

To the best of the author knowledge there are just a few studies on the behavior of evolutionary systems that employ complexity perspectives (e.g. [15],[2],[16], [13]). Still, Kolmogorov complexity and PLS complexity might provide new insights onto the nature of evolutionary search in general and memetic algorithms in particular. The overall strategy we would like to expose in this paper (and propose as the subject matter of future research) is this:

First, the use of Polynomial local search complexity theory to unmask and make explicit the relationships between algorithms (i.e. different Memetic algorithms) with their component parts (i.e. mutation, crossover, selection and local search operators) and the **problems** they are applied to. In particular, to help find **worst case** complexities of the associations between algorithms and problems. Secondly, to use Kolmogorov complexity theory to unmask and make explicit the relationships between algorithms and the problem **instances** they are applied to. Specifically, to provide **best case** complexities for the associations between algorithms and (classes of) instances.

2.1 PLS theory and Memetic Algorithms

A Local Search Problem(LSP) Π is a 4−tuple $\Pi = (D_\Pi, S_\Pi, f_\Pi, N_\Pi)$ where:

D_Π is a set of instances, e.g., Euclidean planar distance matrices for the TSP; $S_\Pi(x)$ is a set of solutions to instance $x \in D_\Pi$, e.g., a set of TSP tours; $f_\Pi(s,x)$ is a cost function for $s \in S_\Pi(x)$ and $x \in D_\Pi$, e.g., the length of tours for TSP instances; $N_\Pi(s,x)$ is a neighborhood function that assigns to every $s \in S_\Pi(x)$ and $x \in D_\Pi$ a set of solutions, $\{n_s\}$, with $n_s \in S_\Pi(x)$.

In [5] we analysed the PLS complexity of the following LSPs:

1. $\Pi_0 = (TSP, S'_{TSP}, f'_{TSP}, N_{TSP,0,M,R})$
2. $\Pi_1 = (TSP, S'_{TSP}, f'_{TSP}, N_{TSP,1,M,R,L})$
3. $\Pi_2 = (TSP, S'_{TSP}, f'_{TSP}, N_{TSP,2,M,R,L})$
4. $\Pi_3 = (TSP, S'_{TSP}, f'_{TSP}, N_{TSP,3,M,R,L})$

In these LSPs the set of instances D_Π is the set of instances of the *TSP*; the set of solutions to $x \in D_{TSP}$, S'_{TSP}, is a (multi)set of tours (also called a population) which are solutions to x. The cost function must be accommodated to account for sets of solutions and this is done by letting $f'_{TSP}(s,x) = min_{t \in s}(tourLength(t))$. This cost function assigns to a population of tours the cost of the best tour in the (multi)set. The crucial part of the definition comes from the appropriate specification of the neighborhoods used in these Local Search Problems. As we intend to model Memetic algorithms which use a mutation operator m, a crossover operator c and (possibly) a local searcher l, we need to include them in our definition of neighborhood. In our analysis, $N(.)$ is the set of all possible outcomes of mutating, crossing over or local searching solutions from an initial population; the numbers 0,1,2,3 are indexes into an architectural taxonomy of memetic algorithms which indicate the sequence in which these operators are scheduled. In [5] we proved that for a variety of commonly (and in practice successfully) employed mutation, crossover and local searcher operators for the *TSP* the resulting $N(.)$ is of polynomial size, hence $\Pi_0, \Pi_1, \Pi_2, \Pi_3$ can be shown to be in PLS. We also proved in [5] that, in fact, $\Pi_0, \Pi_1, \Pi_2, \Pi_3$ are PLS-complete by giving reductions from $\Pi_{TSP-1} = (TSP, S_{TSP}, f_{TSP}, K-Opt)$ [6] and $\Pi_{TSP-2} = (TSP, S_{TSP}, f_{TSP}, Lin-Kernighan)$ [12][1].

[1]With $f(.)$ the usual tour length.

2.2 Kolmogorov theory and Memetic algorithms

The previous section shows that for a range of genetic and local search operators and a variety of ways of scheduling their operations (when trying to solve *TSP* instances) the resulting local search problems are PLS-complete. This implies that under the right circumstances the Memetic Algorithms might take at least exponential time to reach a local optima. We would like to investigate lower bounds for particular instances of the combinatorial problem rather than the whole set of instances. The particular framework we use for that purpose is the following.

A MA is seeing as a particular partial function that operates on two inputs, I and P, and produces an output S: $MA(I,P) = \Phi_I(P) = S$, where I is an instance of a problem (e.g. an instance of the *TSP*) and P is the sequence of random numbers given to the Memetic Algorithm[2] to produce a target solution S to the problem instance I. We saw that for some instances the algorithm might take exponential time to converge into a local optima. Thus $|P|$, the length of the random numbers required (interpreted here as a program for the particular partial function realized by the Memetic Algorithm) will be exponentially long as well, as the neighborhoods used in the definition of the LSPs are all polynomially large (in the size of the instance). That is, each iteration of the algorithm uses only a polynomially long sequence of random numbers but could be iterated an exponentially long number of times. The natural question that arises is which is the Kolmogorov complexity of solution S under the specific partial function implementation Φ_I, that is which is $K_{\Phi_I}(S)$?. That number gives the length of the shortest sequence of random numbers required by the Memetic Algorithm to specify S. That is, the shortest the program P is for S under the particular computing device MA, the faster that target solution can be generated by the device. Although non-computable in principle, the (approximate) estimation of the Kolmogorov complexity of solutions, $K_{\Phi_I}(S)$, can provide valuable insight on how quick can we expect to solve a problem instance with a particular Memetic Algorithm.

2.3 Discussion on Designability

Knowing $K_{\Phi_I}(S)$ however is not enough to estimate how easy it is for Φ_I to produce S. We need to know how many of these shortest programs (or other programs) can generate S. The larger the number of shortest programs P for $MA(I,.)$, the easier will be to produce S. This can be formalized with the concept of "designability" commonly used in studies of protein folding[8][14]. We parameterize the designability $d(.)$ with t (as the length of a program for Φ_I):

$d(S)_t = |\{P \; such \; |P| = t \; and \; \Phi_I(P) = S\}|$ and we identify $d^* = d(S)_{K_{\Phi_I}(S)} = |\{P \; such \; |P| = K_{\Phi_I}(S) \; and \; \Phi_I(P) = S\}|$

In figure 2 we plot an ideal designability distribution for an hypothetical solution to an instance of a combinatorial problem . If the density of programs by which Φ_I can pro-

[2]Remember that a Memetic Algorithm relies on randomization.

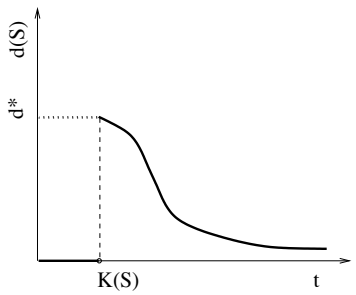

Figure 2: Designability of a solution as a function of the length of random sequences that generates it.

duce S decreases very fast as the length of the programs increases, it becomes more and more difficult to find long programs for Φ_I that produce S, on the other hand, it becomes easier to produce short (i.e. efficient) programs that generates S. The ideal scenario for any specific search strategy is that the distribution of programs (i.e. sequence of random numbers) it needs to produce a given solution, follows as close as possible the distribution depicted if fig. 2 with a maximum at the Kolmogorov complexity[3]. If the curve had the tail to the left then the MA would tend to take very long time to solve an instance as the solutions for that particular instance will usually have very long programs. Thus, the problem of designing efficient searchers becomes a multi-objective problem were one tries to minimize the Kolmogorov complexity of solutions while at the same time tries to maximize the number of programs with minimum length. To characterize searchers and instances we can define a "reachability" factor as $\rho = \frac{t}{d(S)_t}$. If ρ could be minimized for S (or for a set of solutions) then we would have produce an efficient Memetic algorithm[4]. Although we would like ρ to have a minimum at $\rho^* = \frac{K_{\Phi_I}(S)}{d^*(S)}$, cases will exist were it can actually be minimized by having long but numerous programs.

3 Conclusions

In this paper we present very simple arguments on the PLS-Completeness of Memetic Algorithms for the TSP. These complexity results implies that potentially very long path to local optima exists even when the neighborhood used are of polynomial size[5]. These long path can be interpreted as the sequence of random numbers the MA needs in order to generate a target solution. That sequence can then be re-interpreted as a program. The Kolmogorov complexity of solutions based on the specific computation device used (i.e. the MA) gives the length of the shortest such sequence we can expect to produce a solution. In order to construct efficient search algorithms the designer needs to minimize the reachability factor ρ.

Acknowledgments Parts of this work were done during the author's visit to *Centro de Ciencias Matematicas. Universidade Da Madeira*, Madeira, July/August 2002.

[3]Please note the discontinuity of the distribution.

[4]Please note that in general this applies to other search methods as well.

[5]Remember that by definition of the class PLS, these LSPs have polynomial time components.

References

[1] T. Back, D.B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP publishing Ltd and Oxford Unviersity Press, 1997.

[2] E.B. Baum, D. Bone, and C. Garret. Where genetic algorithms excel. *Evolutionary Computation*, 9(1):93–124, 2001.

[3] W. E. Hart. Adaptive global optimization with local search. *Ph.D. Thesis, University of California, San Diego*, 1994.

[4] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. How easy is local search. *Journal of Computer And System Sciences*, 37:79–100, 1988.

[5] N. Krasnogor. http://slater.chem.nott.ac.uk/ ~natk/public/papers.html. In *Studies on the Theory and Design Space of Memetic Algorithms*. Ph.D. Thesis, University of the West of England, Bristol, United Kingdom., 2002.

[6] M.W. Krentel. Structure in locally optimal solutions. In *30th Annual Symposium on Foundations of Computer Science*, pages 216–222. IEEE Computer Society Press, Los Alamitos, CA, 1989.

[7] M.W.S. Land. Evolutionary algorithms with local search for combinatorial optimization. *Ph.D. Thesis, University of California, San Diego*, 1998.

[8] H. Li, R. Helling, C. Tang, and N. Wingreen. Emergence of preferred structures in a simple model of protein folding. *Science*, 1996.

[9] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

[10] P. Merz. *Memetic Algorithms for Combinatorial Optimization Problems:Fitness Landscapes and Effecitve Search Strategies*. Ph.D. Thesis, Parallel Systems Research Group. Department of Electrical Engineering and Computer Science. University of Siegen., 2000.

[11] P. A. Moscato. On evolution, search, optimization, genetic algorihtms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Caltech, Pasadena, California, 1989.

[12] C.H. Papadimitriou. The complexity of the lin-kernighan heuristic for the traveling salesman problem. *SIAM Journal on Computing*, 21:450–465, 1992.

[13] R. P. Salustowicz and J. Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997.

[14] C. Tang. Simple models of the protein folding problem. *Physica A*, 2000.

[15] P.M.B. Vitany. A discipline of evolutionary programming. *Theoretical Computer Science*, 2000.

[16] I. Wegener, J. Scharnow, and K. Tinnefeld. Fitness landscapes based on sorting and shortest paths problems. In *Proceedings of the Parallel Problem Solving from Nature VII. Lecture notes in computer science*, 2002.

[17] M. Yannakakis. Computational complexity. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–55. John Wiley & Sons Ltd., 1997.