Genetic Local Search Algorithms for the Traveling Salesman Problem

Nico L.J. Ulder [1], Emile H.L. Aarts [2,3], Hans-Jürgen Bandelt[4]
Peter J.M. van Laarhoven [5], and Erwin Pesch [4]

1. Océ-Nederland BV, P.O. Box 101, NL-5900 MA Venlo
2. Philips Research Laboratories, P.O. Box 80.000, NL-5600 JA Eindhoven
3. Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven
4. University of Limburg, P.O. Box 616, NL-6200 MD Maastricht
5. Centre for Quantitative Methods, Nederlandse Philips Bedrijven BV,
P.O. Box 218, NL-5600 MD Eindhoven

Abstract

We briefly review previous attempts to generate near-optimal solutions of the Traveling Salesman Problem by applying Genetic Algorithms. Following the lines of Johnson [1990] we discuss some possibilities for speeding up classical Local Search algorithms by casting them into a genetic frame. In an experimental study two such approaches, viz. Genetic Local Search with 2-Opt neighbourhoods and Lin-Kernighan neighbourhoods, respectively, are compared with the corresponding classical multi-start Local Search algorithms, as well as with Simulated Annealing and Threshold Accepting, using 2-Opt neighbourhoods. As to be expected a genetic organization of Local Search algorithms can considerably improve upon performance though the genetic components alone can hardly counterbalance a poor choice of the neighbourhoods.

1 Introduction

Genetic Algorithms (GAs) have been designed as general purpose search strategies and optimization methods; the GA catechism is laid down in Goldberg [1989a] and [1989b]. The very name, though, might be misleading since the word "algorithm" alludes to a special method of solving a certain kind of problem, but what is really meant is a general strategy or metaheuristic calling on principles of evolution.

Roughly speaking, a Genetic Algorithm aims at producing near-optimal solutions by letting a set of random solutions undergo a sequence of unary and binary transformations governed by a selection scheme biased towards high-quality solutions. Rechenberg [1973] had already experimented with a kind of Genetic Algorithm dubbed "erweiterte Evolutionsstrategie". The so-called "Evolutionary Strategies" now usually refer to methods not embodying bit-string recombination, or crossover operators; see Schwefel [1977] for an exposition of pertinent optimization techniques and Ablay [1987] for an application to the TSP.

Problems from combinatorial optimization are well within the scope of Genetic Algorithms, so it was inevitable that the Traveling Salesman eventually became a victim of GA activities. Early attempts closely followed the scheme of what Goldberg [1989a] now called a Simple GA and were actually rather discouraging when compared with standard TSP heuristics, for instance, the experiments of Grefenstette et al. [1985] lead to solutions as far as 25% from the optimum, in case of a 50-city TSP.

The conclusion, however, that "Genetic Algorithms are not well suited for fine-tuning structures which are very close to optimal solutions" [Grefenstette, 1987] is a bit precipitate. As Suh & Van Gucht [1987] emphasize "it is ... essential if a competitive Genetic Algorithm is desired, to incorporate ... local improvement operators into the recombination step of a Genetic Algorithm". A resulting algorithm has then been called Heuristic GA, which in a way is a pleonasm since every GA incorporates - at least implicit - heuristic information about the problem. Since Lin & Kernighan [1973] the prevalent local improvement operator is a 2-exchange. Jog et al. [1989] further improve their Genetic Algorithm by incorporating Or-exchanges.

Equally essential is the careful selection of the binary recombination operator, the crossover, that entails heuristic information; see again Suh & Van Gucht [1987]. For the TSP, Mühlenbein et al. [1988] propose a binary recombination operator that transplants a subpath of the first tour into the appropriately modified second tour.

In this paper we address the question as to what extent concepts from population genetics can improve the performance of classical Local Search algorithms. For this we concentrate on a numerical study for the TSP in which the performance of Genetic Algorithms is compared with that of more classical search algorithms such as multi-start Local Search, Simulated Annealing and Threshold Accepting. The remainder of the paper is organized as follows. First we give a template of a general Genetic Local Search algorithm and show how it can be tailored to the TSP. Next we describe the setup of our numerical study and present the results that were obtained. The paper is concluded with a discussion of the potentials of Genetic Local Search algorithms, for now and in the future.

2 Genetic Local Search

It is desirable to put the previous approaches to the TSP using GA into appropriate perspective. Every successful strategy to produce near-optimal solutions necessarily relies upon some efficient iterative heuristic, typically a Local Search technique. Well-known Local Search algorithms for the TSP are the 2-Opt algorithm (because of its efficiency), the Lin-Kernighan algorithm (because of its effectiveness), and special variants of k-Opt algorithms (such as the Or-Opt algorithm). All these algorithms differ with respect to their neighbourhood structures. Any such structure specifies a set of neighbouring solutions that are in some sense close to that solution. The associated local improvement operator replaces a current solution by a neighbouring solution of better value if possible. Then Local Search - starting from some initial solution - proceeds by applying this operator until a local optimum is reached. See Johnson et al. [1988] for more information on Local Search and its complexity.

In practice, multi-start Local Search is used rather than a single run, i.e. the Local Search algorithm is repeated several times, retaining the best local optimum found. It is plausible that independent multiple runs of a Local Search algorithm generally will not constitute an effective procedure since, loosely speaking, every individual solution has to find its own way to near-optimal regions. Cooperation and competition

1. **Initialize**: Construct an initial population of solutions.
2. **Improve**: Use a Local Search algorithm to replace each solution in the current population by a better solution, e.g., a local optimum.
3. **Recombine**: Extend the current population by adding solutions obtained by recombining two or more solutions in the current population.
4. **Improve**: Use a Local Search algorithm to replace each offspring solution in the current population by a better solution, e.g., a local optimum.
5. **Select**: Reduce the extended population to its original size according to prescribed selection rules.
6. **Evolve**: Repeat steps 3 to 5 until some stopping criterion is met.

Table 1: *Genetic Local Search.*

between individual solutions should certainly contribute to the overall performance of an algorithm. Several authors have therefore devised a collective organization of Local Search algorithms, drawing ideas from population genetics; see e.g., Ackley [1987], Suh & Van Gucht [1987], Mühlenbein et al. [1987, 1988], Mühlenbein & Kindermann [1989], Mühlenbein [1989], Gorges-Schleuter [1989], Jog et al. [1989]. These approaches can be schematized as is shown in Table 1. This scheme is just a template, which requires further refinements in order to design a successful algorithm. We will now briefly mention a number of options in each step.

As to initialization, one would often generate random populations. At least in the case of the TSP, there is a wealth of tour construction heuristics that could be used to make up an initial population of medium quality; see Lawler et al. [1985], or Johnson [1990].

The Local Search algorithm of choice in the improvement step should simply be the best one available that meets given time capacity constraints. For the TSP this is - beyond doubt - the heuristic due to Lin & Kernighan [1973]. In case that severe time restrictions are imposed one can still use a truncated version of the Local Search algorithm such that it goes through only a small number of iterations.

Besides the, carefully designed binary recombination operators one may also introduce operators of higher arities such as consensus operators, that fix edges common to most TSP tours of a current population; see Mühlenbein [1989], cf. the reduction procedure of Lin & Kernighan [1973].

Selection can be realized in a number of ways: one could adopt the scenario of Goldberg [1989a] or use deterministic ranking. Further it matters whether new recombined offspring solutions compete with the parent solutions or simply substitute them. A promising modification of recombination and selection involves the design of a population structure that defines proximity between positions of individuals, resulting in overlapping cliques, called demes. Then recombination and selection is restricted to take place only among the individuals from each deme; see Gorges-Schleuter [1989].

3 Numerical Results

We have tested two basic versions of Genetic Local Search algorithms for the TSP. Both algorithms depart from random populations of solutions, the population sizes being variable and dependent on the problem instances. The first one uses the 2-Opt neighbourhood structure for the Local Search in the improvement step, so that the standard 2-Opt heuristic is performed on each individual tour. The second one uses the more complicated Lin-Kernighan neighbourhood structure, thus yielding a pair of improvement operators, viz., the dynamical k-exchange and the additional 4-exchange as described in the original paper of Lin & Kernighan [1973]. We adopted the implementation due to Lageweg (CWI Amsterdam), disregarding the optional reduction part.

In both algorithms recombination is done by taking two tours at random in the current population and implanting a carefully chosen subpath of one of the tours - containing at most one third of all cities - into the other one, in essentially the same way as was proposed by Mühlenbein et al. [1988] and Gorges-Schleuter [1989].

Selection is executed by simply collecting the best tours of the extended population. The algorithm stops when either all tours in the current population have the same length or the length of the best tours did not improve within five successive generations.

We compared the performance of the above two algorithms with that of the corresponding multi-start Local Search algorithms, as well as with Simulated Annealing (SA) and its deterministic variant Threshold Accepting (TA) due to Dueck & Scheuer [1988]. Both SA and TA use the 2-Opt neighbourhood structure. For Lin-Kernighan and TA the original FORTRAN code was translated to PASCAL (in a straightforward manner), so that all six programs were in PASCAL. Moreover, care was taken to have identical data structures and subroutines wherever possible. Our experimental study is based on a comparison of the statistical averages of the tour lengths of the final solutions obtained by applying the six algorithms five times each to eight well-known instances of the TSP, ranging from 48 up to 666 cities. For each instance, the algorithms are all allowed an almost equal amount of running time. So we focus on effectiveness rather than efficiency. The reference points are given by SA according to the cooling schedule of Aarts & Van Laarhoven [1985], with the parameter value $\delta = 1$. In order to have the stopping criterion for the two Genetic Local Search algorithms fulfilled just within the time bounds provided by each run of SA, we adjusted the free parameter, the population size, accordingly. Indeed, the larger the populations become, the more diversity we get and thus longer running times. Table 2 gives the average deviations from the known optimal solutions. The genetic versions Gen2-Opt and GenLK of 2-Opt and Lin-Kernighan, respectively, perform clearly better than their multiple-run companions. Moreover, GenLK is superior to the other algorithms. In contrast to the 2-Opt and the LK variants, the outcomes for SA and TA do not change considerably with the problem sizes; the average deviations from an optimum are 2.4% for SA and 2.0% for TA over all instances.

Now, let us have a closer look at the numbers of iterations (trials or runs) that were needed to arrive at the solutions from Table 2. It is interesting to compare Mult2-Opt and Gen2-Opt - and the two LK versions - in this respect. See Table 3: Gen2-Opt allows 3.2 to 9.2 more single runs of 2-Opt than Mult2-Opt - the corresponding numbers for LK

Instance	$\bar{t}$	SA	TA	Mult2-Opt	MultLK	Gen2-Opt	GenLK
GRO48	6	1.89	1.65	1.35	0	0.19	0
TOM57	10	1.94	2.88	1.34	0	0.50	0
EUR100	60	2.59	3.41	3.23	0	1.15	0
GRO120	86	2.94	2.01	4.57	0.08	1.42	0.05
LIN318	1600	2.37	1.27	6.35	0.37	2.02	0.13
GRO442	4100	2.60	1.31	9.29	0.27	3.02	0.19
GRO532	8600	2.77	1.79	8.34	0.37	2.99	0.17
GRO666	17000	2.19	1.70	8.67	1.18	3.45	0.36

Legend to the table:

$\bar{t}$	:	Average running time in seconds on a VAX 8650 under VMS 5.1
SA	:	Simulated Annealing with 2-Opt neighbourhoods
TA	:	Threshold Accepting with 2-Opt neighbourhoods
Mult2-Opt	:	Multi-start Local Search with 2-Opt neighbourhoods
MultLK	:	Multi-start Local Search with Lin-Kernighan neighbourhoods
Gen2-Opt	:	Genetic Local Search with 2-Opt neighbourhoods
GenLK	:	Genetic Local Search with Lin-Kernighan neighbourhoods
GRO48	:	instance with 48 cities due to Grötschel
TOM57	:	instance with 57 cities due to Karg & Thompson
EUR100	:	instance with 100 cities due to Aarts & Van Laarhoven
GRO120	:	instance with 120 cities due to Grötschel
LIN318	:	instance with 318 cities due to Lin & Kernighan
GRO442	:	instance with 442 cities due to Grötschel
GRO532	:	instance with 532 cities due to Grötschel
GRO666	:	instance with 666 cities due to Grötschel

Table 2: *Performance comparison of six Local-Search-based algorithms: average relative deviation from the optimal tour length in % for eight well-known instances of the Traveling Salesman Problem.*

are 1.2 and 2.7, respectively - although the genetic variants still have to spend additional time on recombination and selection. It thus pays off when intermediate solutions are of higher quality. Note that the population sizes forced by the experimental design are quite small: for GenLK the sizes range from 8 to 10 while the range is 14 to 56 in the case of Gen2-Opt.

Similar experiments have been carried out for other time conditions, for example using $\delta = 0.1$ in the cooling schedule of the SA algorithm. The obtained results show a similar behaviour as those obtained for $\delta = 1$. Furthermore, we have run experiments where heavy time constraints were imposed. Under such circumstances truncations of 2-Opt or Lin-Kernighan were needed in the improvement step in order to achieve satisfactory results. The population sizes ranged from 10 to 30 and were thus smaller than recommended for Simple GAs design by Grefenstette [1986].

Instance	SA	TA	Mult2-Opt	MultLK	Gen2-Opt	GenLK
GRO48	89,112	180,000	40	19	140	24
TOM57	145,236	300,000	41	14	140	32
EUR100	668,250	1,800,000	67	31	216	40
GRO120	1,135,260	2,400,000	64	30	216	48
LIN318	15,221,706	45,000,000	99	37	390	100
GRO442	34,988,499	120,000,000	108	67	720	100
GRO532	61,442,010	246,000,000	116	77	954	120
GRO666	112,494,060	450,000,000	122	45	1,120	100

Table 3: *Total numbers of trials (SA, TA), numbers of single runs (Mult2-Opt, MultLK), and population sizes times generation numbers (Gen2-Opt, GenLK), respectively, for the data in Table 2.*

4 Discussion

To investigate the potentials of Genetic Local Search in combinatorial optimization, we applied two pertinent algorithms to the TSP. The TSP offers a great challenge since there exists a plethora of approximative algorithms for this problem that serve as a good comparative standard. Among these, the algorithms based on exchange heuristics are the widest used ones, with the Lin-Kernighan heuristic as the uncontested champion; cf. Johnson [1990]. We decided to implement Genetic Local Search - incorporating the 2-Opt and Lin-Kernighan heuristics - in a straightforward way: only a single parameter, i.e. the population size, has to be chosen by the user in advance. It is therefore not necessary to first calibrate a whole bunch of parameters in order to get reasonably good solutions. Our experimental study indicates that Genetic Local Search is consistently superior to the notorious multi-start Local Search. With larger problem sizes it becomes apparent that this simplistic strategy tends to strand at local optima of only moderate quality, so that the SA and TA algorithms eventually beat Gen2-Opt in this experiment. Certainly, both Gen2-Opt en GenLK can be improved further by integrating more subroutines inferred from population genetics; see Mühlenbein [1989]. Still, GenLK seems to outperform the "champions" ASPARAGOS and TA on the TSP. This is not really surprising since the much poorer 2-Opt heuristic - or a truncation thereof - is embedded in the latter two algorithms. The additional use of Or-Opt, or Or-exchanges, does not seem to provide strinkingly better results either; cf. Mühlenbein et al. [1988] and Jog et al. [1989]. One word of caution is in order here: there is an ungoing confusion between "proto-heuristic" - Local Search technique and crossover - for a specific problem and "metaheuristic" - general purpose solution strategy- for organizing the particular proto-heuristic effectively. It is thus notoriously spoken of the SA and the TA algorithms for the TSP although either procedure embodies a specific heuristic operator, viz., the 2-exchange. We have refrained from including SA and TA endowed with dynamic k-exchange instead in our experimental study since this would have required extra testing of appropriate cooling schedules and threshold sequences. Anyway, Genetic Local Search should not be viewed as being opposed to SA or TA because elements of these strategies can be implemented in Genetic Local Search at the improvement or selection step. A genetic organization of some basic Local Search algorithms would constitute only one out of many implementation devices that are necessary in order to cope with very large problem sizes; see concluding remarks and announced research in Johnson [1990]. In the case of the TSP with thousands of cities, some hierarchical structuring of the solution strategy seems to be unavoidable; for a pertinent approach commencing by a geometrical clustering of the cities; see Reinelt [1989]. Then Genetic Local Search techniques may enter into any level of a hierarchically organized strategy to further improve intermediate solutions.

Acknowledgement

We would like to thank the other members of the Eindhoven-Maastricht study group on artificial intelligence for stimulating discussions.

References

AARTS, E.H.L. AND P.J.M. VAN LAARHOVEN [1985], A New Polynomial-Time Cooling Schedule, *Proc. IEEE Int. Conf. Computer-Aided Design*, Santa Clara, pp. 206-208.

ABLAY, P. [1987], Optimieren mit Evolutionsstrategien, *Spektrum der Wissenschaft* 7, 162-173.

ACKLEY, D.H. [1987], *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers.

DUECK, G. AND T. SCHEUER [1988], Threshold Accepting. A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *TR 88.10.011*, IBM Heidelberg Scientific Center.

GOLDBERG, D.E. [1989a], *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

GOLDBERG, D.E. [1989b], Zen and the Art of Genetic Algorithms, *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 80-85.

GORGES-SCHLEUTER, M. [1989], ASPARAGOS, A Parallel Genetic Algorithm and Population Genetics, *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan Kaufmann publishers, pp. 422-427.

GREFENSTETTE, J.J. [1986], Optimization of Control Parameters for Genetic Algorithms, *IEEE Trans. Systems Man Cybernetics* 16, 122-128.

GREFENSTETTE, J.J. [1987], Incorporating Problem Specific Knowledge into Genetic Algorithms, in L. Davis, *Genetic Algorithms and Simulated Annealing*, Pitman, pp. 42-60.

GREFENSTETTE, J.J., R. GOPAL, B. ROSMAITA, AND D. VAN GUCHT [1985], Genetic Algorithms for the Traveling Salesman Problem, *Proc. 1st Int. Conf. Genetic Algorithms and Their Applications*, Lawrence Erlbaum Ass., pp. 160-168.

JOG, P., J.Y. SUH, AND D. VAN GUCHT [1989], The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem. *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 110-115.

JOHNSON, D.S. [1990], Local Optimization and the Traveling Salesman Problem, *Proc. 17th Colloq. Automata, Languages, Programming*, Springer-Verlag (in press).

JOHNSON, D.S., C.H. PAPADIMITRIOU, AND M. YANNAKAKIS [1985], How Easy is Local Search?, *Journal of Computer and System Science* 37, 79-100.

LAWLER, E.L., J.K. LENSTRA, A.H.G. RINNOOY KAN, AND D.B. SHMOYS [1985], *The Traveling Salesman Problem*, John Wiley & Sons.

LIN, S., AND B.W. KERNIGHAN [1973], An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Operations Research* 21, 498-516.

MÜHLENBEIN, H. [1989], Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization, *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 416-421.

MÜHLENBEIN, H., M. GORGES-SCHLEUTER, AND O. KRÄMER [1987], New Solutions to the Mapping Problem of Parallel Systems: the Evolution Approach, *Parallel Computing* 4, 269-279.

MÜHLENBEIN, H., M. GORGES-SCHLEUTER, AND O. KRÄMER [1988], Evolution Algorithms in Combinatorial Optimization, *Parallel Computing* 7, 65-85

MÜHLENBEIN, H., AND J. KINDERMANN [1989], Dynamics of Evolution and Learning - Towards Genetic Neural Networks, in R. Pfeiffer, *Connectionism in Perspective*, Elsevier (in press).

RECHENBERG, I. [1973], Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, *Problemata*, Fromann-Holzboog.

REINELT, G. [1989], Fast Heuristics for Large Geometric Traveling Salesman Problems, *Report nr. 185*, Institut für Mathematik, Universität Augsburg.

SCHWEFEL, H.-P. [1977], *Numerische Optimierung von Computer-Modellen Mittels der Evolutionsstrategie*, Birkhäuser Verlag.

SUH, J.Y., AND D. VAN GUCHT [1987], Incorporating Heuristic Information into Genetic Search, *Proc. 2rd Int. Conf. Genetic Algorithms*, Lawrence Erlbaum Associates, pp. 100-107.