Home problem 2

Simon Sigurdhsson
900322–0291

Email Sigurdhsson@gmail.com

Abstract This report discusses the solutions to three out of four problems in home problem set 2 of the course in Stochastic optimization algorithms given by Chalmers University of Technology. In the first part, the Travelling Salesman Problem is considered and solved using a standard genetic algorithm as well as Ant Colony optimization techniques. In the second part, a Particle Swarm optimization algorithm is implemented and applied to a simple minimization problem as well as an integer programming problem. Finally, in the third part, Linear Genetic Programming is used to fit a function to provided data points.

1 Problem 2.1

The first problem concerns the (Euclidean) Travelling Salesman Problem, in which the shortest Hamiltonian cycle[1] of a complete, weighted graph is sought. A data set with 50 nodes of a complete graph, and their positions (from which the graph weights may be obtained) has been provided.

The problem is NP-complete,and therefore no deterministic algorithm of sub-exponential time complexity exists. It will be shown that the number of solutions to TSP is very large (on the order of $3 \cdot 10^{62}$ for 50 nodes), and as such a brute-force minimization on the complete search space is out of the question. Therefore, two different stochastic optimization algorithms will be applied to the problem.

First, a simple genetic algorithm seeded with a completely random population is implemented and applied to the problem instance. Then, the ant colony optimization algorithm is introduced. Finally, the genetic algorithm is modified to use an initial population based on the greedy nearest-neighbour solution of TSP.

1.1 Part A

In order to demonstrate the difficulty of TSP, which will be solved using evolutionary algorithms and ant colony optimization below, an upper bound[2] on the number of possible tours given N cities is established.

Assuming a complete graph (*i.e.* there is an edge between every pair of nodes i, j there are $N!$ permutations of a complete enumeration of the nodes, and as such there are $N!$ tours (paths that visit all nodes once and only once).

Assuming now that cyclic permutations are equivalent, and that the direction

1. A path visiting all nodes of the graph exactly once.
2. The bound is an upper bound due to the assumption of full connectivity, which holds for the problem as defined here but not in the more general case.

of a tour is irrelevant, the number of distinct tours are reduced by a factor $2N$. This yields a final count of $\frac{1}{2}(N-1)!$ possible tours.

1.2 *Part B*

The first algorithm implemented to solve TSP is a fairly standard genetic algorithm, similar to that implemented in the first home problem (Sigurdhsson 2013). It deviates from the standard genetic algorithm described by Wahde (2008, p. 56) in only a few respects. First, crossover is entirely neglected (equivalent to setting $p_c = 0$) to simplify the handling of chromosomes, which are taken to represent complete cycles of the graphs (each gene representing a node in the sequence). Additionally, mutations are implemented as swap mutations (Wahde 2008, p. 147).

The algorithm was run with $p_{\text{mut}} = \frac{1}{51}$, $p_{\text{tour}} = 0{,}750$, tournament size $j = 2$, population size $N = 50$ and 10 000 iterations. Since each chromosome represented a full tour of the graph, the chromosome length remained fixed at 51. The tour found using these settings had length 174,060, and is shown in fig. 1.

1.3 *Part C*

The simple genetic algorithm works well, but isn't adapted to the problem at hand. Using ant colony optimization, which is inspired by the path-finding properties of ants (Wahde 2008, pp. 99 sqq.), better performance may be obtained. The basic shell of the ant colony optimization was provided in `AntSystem.m`, with clear specifications of the implementation of specific functionality in external functions.

The algorithm was applied to the given problem using parameters $\alpha = 1$, $\beta = 1$, $\rho = \frac{1}{2}$ and 50 ants. After approximately 250 iterations, the tour of length 121,340 shown in fig. 3 was found.

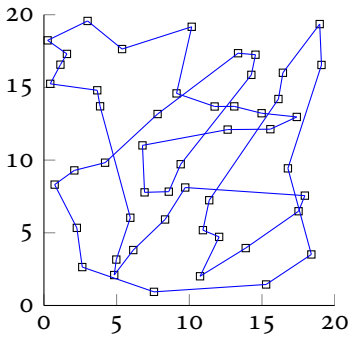

Figure 1: *Shortest tour found after* 10 000 *iterations of the first genetic algorithm.*

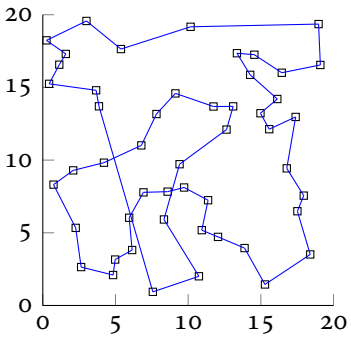

Figure 2: *Shortest tour found after* 10 000 *iterations of the second genetic algorithm.*

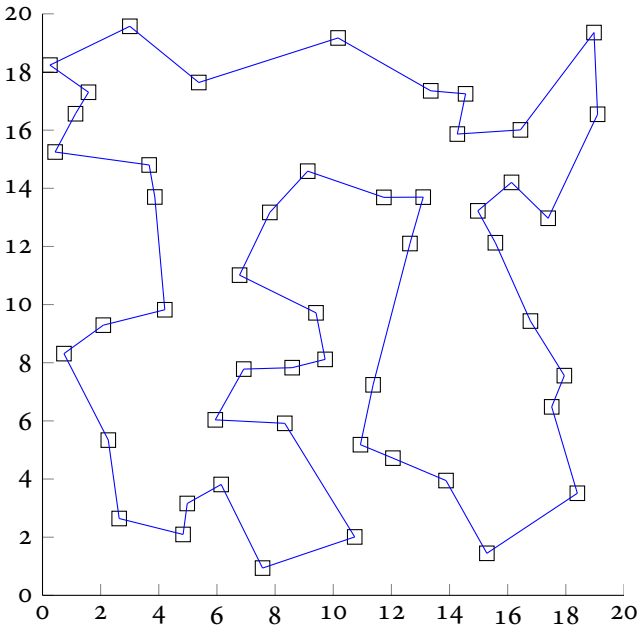

Figure 3: *Shortest tour found after* ≈ 250 *iterations of ant colony optimization.*

1.4 Part D

Part of the performance gained from using ant colony optimization might result from the heuristic applied in the initial stages of the algorithm, where a greedy nearest-neighbour path is used to seed the initial pheromone levels. In order to have a fair comparison, a a similar heuristic for the genetic algorithm is of interest.

The genetic algorithm discussed earlier was modified so that the initial population, instead of being randomly generated, instead was taken as a greedy nearest-neighbour tour with random swap mutations. The algorithm was applied to the problem with identical parameters, and the shortest tour found after 10 000 iterations (shown in fig. 2) has length 133,920.

2 Problem 2.2

The second problem is centered around the implementation and testing of the particle swarm stochastic optimization method. A standard particle swarm optimization was implemented as described by Wahde (2008, p. 123), including an inertia weight (Wahde 2008, p. 128). This algorithm was then applied to two separate unconstrained[3] minimization problems: one general optimization problem in two variables and one integer programming problem in five variables.

3. The minimization problems as presented were unconstrained, with instructions to limit the search space.

Table 1: *Parameters used by the (standard) particle swarm optmization algorithm applied to the first minimization problem.*

Parameter	$v_{\max}$	α	Δt	-1	1	$w^{\star}$	$\beta^{\dagger}$	$w_{\text{low}}{}^{\ddagger}$
Part A	0,50	1,00	1,00	2,00	2,00	1,40	0,99	0,40
Part B	1,00	1,00	1,00	2,00	2,00	1,40	0,99	0,40

* Initial inertia weight. † Inertia weight factor. ‡ Lower inertia weight.

2.1 Part A

The first problem applied to the particle swarm optimization implementation was the unconstrained optimization problem to minimize

$$f(x, y) = 1 + (-13 + x - y^3 + 5y^2 - 2y)^2 + (-29 + x + y^3 + y^2 - 14y)^2.$$

The search space was restricted to $(x, y) \in [-10, 10]^2$, and the optimization was performed with 100 particles over 10 000 iterations. The parameters used by the algorithm were set as shown by table 1.

The algorithm was run 25 times, finding the best global minimum at $\boldsymbol{x}^* = (4{,}193, 4{,}019)$, with $f(\boldsymbol{x}^*) = 1$, in all of the runs.

2.2 Part B

The second problem applied to the algorithm was an integer programming problem in five variables, to minimize the function

$$f(\boldsymbol{x}) = -\begin{pmatrix} 15 & 27 & 36 & 18 & 12 \end{pmatrix} \boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}} \begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 31 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix} \boldsymbol{x}.$$

The search space was restricted to $\boldsymbol{x} \in [-30, 30]^5 \subset \mathbb{Z}^5$.

Implementation-wise, a few modifications were made to adapt the algorithm to the integer restriction. First, particle positions were rounded after initialization, and before evaluating particles. In addition, the best partcile and swarm positions were saved in their rounded format, while actual particle positions were kept unrounded between iterations.

The algorithm was run with 300 particles over 1000 iterations, with parameter values as shown in table 1. This was done 25 times, yielding a best global minimum at $\boldsymbol{x}^* = (0, 12, 23, 17, 6)$ with function value $f(\boldsymbol{x}^*) = -737$. The average function value found by the algorithm was $f(\boldsymbol{x}) = -721{,}050$ with standard deviation 15,659.

Table 2: *Parameters used by the (standard) LGP algorithm applied to the problem. The seven constant registers were defined as* $c_i \in \{-1, 1, 2, 3, 5, 7, 10\}$*, and the run was limited to* 50 000 *generations.*

Parameter	p_{tour}	p_{mut}	p_c [*]	j [†]	n [‡]	M [§]	N [¶]
Value	0,75	0,025	0,20	100	5	6	7

[*] Crossover probability. [†] Tournament size. [‡] Population size.
[§] Variable registers. [¶] Constant registers.

3 Problem 2.4

The last problem concerns the fitting of a function to given data using linear genetic programming. The data is assumed to be of the form

$$g(x) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p}{b_0 + b_1 x + b_2 x^2 + \cdots + b_q x^q}, \tag{1}$$

i.e. the quotient of two polynomials.

A linear genetic program routine was implemented, based on the standard genetic algorithm discussed in the first home problem and the course literature (Sigurdhsson 2013; Wahde 2008, p. 56), including crossover, mutation and elitism. The major algorithmic differences consist of a modified crossover function, which implements two-point crossover adapted to the specific length of an LGP instruction (so as not to split instructions when performing crossover), and a restricted mutation method that keeps track of what values are allowed at specific points in each instruction of a chromosome.

The parameters used are shown in table 2. Notable differences to earlier results (Sigurdhsson 2013, pp. 6–9) are the increased number of tournament selection rounds and the low crossover probability, both rejected by Sigurdhsson (2013). It is not unreasonable to expect optimal parameter values to differ when applied to dissimilar problems, as is the case here, and one could for instance motivate the lower crossover probability with the fact that crossover has greater potential to "destroy" a LGP than it has to *e.g.* destroy information in binary-encoded variables.

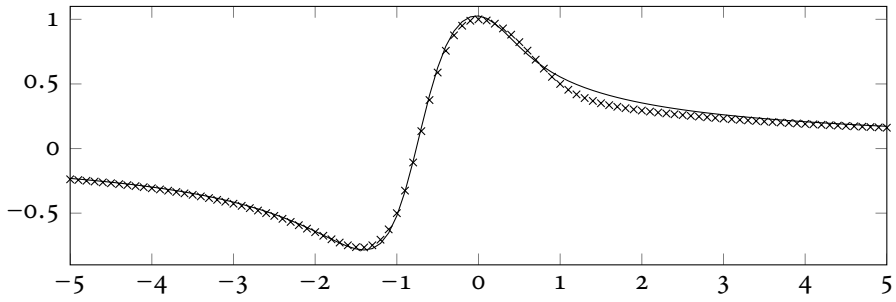

Figure 4: *Best approximation (solid line) of $g(x)$, found after 42 612 iterations of the LGP algorithm, compared to given data (crosses). The approximation has error $e = 0{,}0315$ when compared to the given data.*

The algorithm was seeded with an initial population consisting of chromosomes whose length varied from 10–50, and was left to run for 31 663 generations. The resulting optimal LGP is shown in fig. 4, and has error $e = 0{,}0315$. The LGP consists of 42 instructions (24 of which are effective, *i.e.* contribute to the output), uses all of the six available variable registers and five of the constant registers. When translated to the form given in eq. (1) it defines the function

$$\hat{g}(x) = \frac{1680 + 1197x + 1701x^2 + 3969x^3}{1637 + 1269x + 3816x^2 + 4725x^3 + 3969x^4}.$$

References

Sigurdhsson, S. 2013. "Home problem 1". Submitted as solution to the first problem set of *FFR105 Stochastic optimization algorithms*. Göteborg, Sweden, 27th September.

Wahde, M. 2008. *Biologically Inspired Optimization Methods: An Introduction*. 1st ed. Southampton, Boston: WIT Press. ISBN: 978-1-84564-148-1.