A New Genetic Algorithm for the Optimal Communication Spanning Tree Problem

Yu LI and Youcef BOUCHEBABA

LaRIA, Univ. de Picardie Jules Verne
33, Rue St. Leu, 80039 Amiens Cedex, France
fax: (33) 3 22 82 75 02
e-mail: {yli@laria.u-picardie.fr, boucheba@laria.u-picardie.fr}

Abstract. This paper proposes a new genetic algorithm to solve the Optimal Communication Spanning Tree problem. The proposed algorithm works on a tree chromosome without intermediate encoding and decoding, and uses crossovers and mutations which manipulate directly trees, while a traditional genetic algorithm generally works on linear chromosomes. Usually, an initial population is constructed by the standard uniform sampling procedure. But, our algorithm employs a simple heuristic based on Prim's algorithm to randomly generate an initial population. Experimental results show that our genetic algorithm is simple and efficient to get an optimal or near-optimal solution to the OCST problem.

1 Introduction

A key issue for designing a genetic algorithm (GA) is how to represent the solution space. A traditional GA (e.g. the Simple Genetic Algorithm of Goldberg (1989)) generally uses a fixed representation space, which consists of binary strings or vectors, called *linear chromosome*. Linear chromosome is natural for problems defined directly over binary strings or vectors. However, for problems in which possible solutions are more complicated objects, e.g. the Optimal Communication Spanning Tree (OCST) problem addressed in this paper, for which a solution is a tree, a linear chromosome may be unnatural or even ineffective.

Several researchers have felt the importance of incorporating domain knowledge into the representation of the solution space (Radcliffe & Surry 1995; Wolpert & Macready 1995). Only one practical approach, to our best knowledge, was developed by Sinclair (1998) for optical network optimization, modeling and design (NOMaD), in which network objects are themselves the structures undergoing adaptation and a variety of specific-problem operators have been incorporated within NOMaD.

Although linear chromosome seems to unnatural or even ineffective for incorporating domain knowledge into GAs, there is still a continued interest to follow this traditional framework of GAs in some ways. This is the case in the field of tree network design. Several tree encodings have been suggested to code a spanning tree, such as Predecessor (Berry *et al.* 1995) , Prüfer number (Julstrom

1993), a tree encoding based on determinant factorisation (Abuali *et al.* 1995), and a more complicated chromosome encoding involving bias values for nodes and links (Palmer & Kershenbaum 1995). The advantages and disadvantages of these encodings have been discussed in detail in their papers.

Another important issue for designing a GA is the initialization procedure. In a traditional GA, an initial population is chosen uniformly at random from the whole search space. However, all GA practitioners have observed that a bad initialization procedure may, in the best case, modify the online performance (i.e. increase the time-to-solution), and, in the worst case, prevent the convergence towards the global optimum.

This important issue drew the attention of some researchers (Surry & Radcliffe 1996; Kallel & Schoenauer 1997). It is argued that the standard uniform sampling procedure could be improved by incorporating the domain knowledge into the initialization procedure.

In this paper, we propose a new GA to solve the OCST problem, in which the tree structure itself is taken as the chromosome without intermediate encoding and decoding. The crossovers and mutations directly manipulate trees, and an initial population is generated randomly by a simple heuristic based on Prim's algorithm rather than uniformly at random in the whole solution space. Experimental results show that our GA is simple and effective to get an optimal or near-optimal solution to the OCST problem.

Our experiences suggest that the GA could work directly on the representations which capture the structure of problems rather than on a traditional linear chromosome. We also feel that an initial population of "good quality"could greatly improve the performance of GAs.

The paper is organized as follows. A formal definition of the OCST problem is given in Section 2. Section 3 presents our tree chromosome, the crossover, and mutation operators. Section 4 describes the initialization procedure. We give an overview of our GA in Section 5. The numerical results are presented in the same section. We make some concluding remarks in Section 6.

2 Problem Definition

The problem of finding a spanning tree which minimizes the cost of transmitting a given set of communication requirements between n sites over the tree edges was introduced by Hu (1974) and referred to as OCST problem. It was also referred to as Minimum Communication Cost Spanning Tree problem (see Crescenzi *et al.* 1998). This problem arises directly in the design of tree communication networks. It was proved NP-complete by Johnson *et al.* (1978).

Formally, we are given a complete undirected graph $G = (V, E)$, the distance $d_{i,j}$ between each pair of nodes, and the communication requirement $r_{u,v}$ for each pair of nodes. We assume that the corresponding requirement matrix R is symmetric or uppertriangular, and that the corresponding distance matrix D is symmetric or uppertriangular, but it need not obey the triangle inequality. We wish to find a spanning tree of minimum communication cost.

The communication cost of a given spanning tree is defined as follows. For a pair of nodes u and v, there is a unique path in the spanning tree between u and v. The distance of the path is the sum of distances of edges in the path. The communication cost for the pair of nodes u and v is $r_{u,v}$ multiplied by the distance of the path. Summing over all $\binom{2}{n}$ pairs of nodes, we have the cost of spanning tree. The objective function is then

$$\min_{T \in \Im} \left[\sum_{u,v \in V} (r_{u,v} \sum_{(i,j) \in P_{u,v}(T)} c_{i,j}) \right]$$

where $\Im$ is the set of n^{n-2} labelled trees on n nodes (Gibbons 1985), and $P_{u,v}(T)$ is the unique path between u and v in the tree T.

For example, given a network of six nodes, the distances and the requirements between the six nodes are as follow

$$\mathcal{D} = \begin{pmatrix} 0 & 3 & 6 & 5 & 9 & 7 \\ 0 & 0 & 3 & 2 & 4 & 8 \\ 0 & 0 & 0 & 3 & 7 & 2 \\ 0 & 0 & 0 & 0 & 9 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathcal{R} = \begin{pmatrix} 0 & 5 & 13 & 12 & 8 & 9 \\ 0 & 0 & 7 & 4 & 2 & 6 \\ 0 & 0 & 0 & 3 & 10 & 15 \\ 0 & 0 & 0 & 0 & 11 & 7 \\ 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The cost of the spanning tree in Figure 1, $cost(T)$, is

$$\begin{aligned} cost(T) &= r_{12}(d_{12}) + r_{13}(d_{12} + d_{24} + d_{46} + d_{63}) + r_{14}(d_{12} + d_{24}) + ... + r_{46}(d_{46}) + r_{56}(d_{56}) \\ &= 5(3) + 13(3 + 2 + 2 + 2) + 12(3 + 2) + ... + 7(2) + 12(1) \\ &= 534 \end{aligned}$$

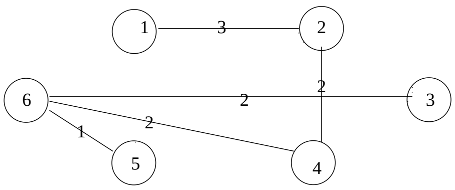

Fig. 1. A spanning tree of six nodes

Note that this problem is different from the familiar Minimum Spanning Tree (MST) which is computationally easy (algorithms of Kruskal, Prim).

Hu (1974) gave exact solutions for two restricted cases of this problem, and Peleg (1998) provided approximate solution to several other cases of the problem. However, for the general case of the problem, we only find two GAs (Palmer & Kershenbaum 1995; Berry *et al.* 1995) to obtain near-optimal solutions.

3 Tree chromosome, crossover and mutation

We use tree itself as chromosome in our GA. An individual in the population is then a tree. This chromosome representation requires new crossovers and mutations to directly manipulate trees.

3.1 Crossover

By blending traditional uniform crossover with tree chromosome representation, we obtain two crossovers. One is based on path exchange called *path crossover*, and another on subtree exchange called *tree crossover*.

Path crossover firstly generates k random paths in two trees (parents) (k is a random number), then exchanges these paths between two trees by inserting a path of a tree into another tree and at the same time deletes cycles produced during the insertion.

The procedure of *path crossover* is described in Figure 2.

```
Procedure pathCrossover(parent1, parent2, child1, child2)
Begin
   choose a random number k
   for i = 1 to k do
   begin
      path1 = randomPath(parent1)
      path2 = randomPath(parent2)
      insertPath(path1,child2)
      insertPath(path2,child1)
   end
End.
```

Fig. 2. Path crossover procedure

An example is given in Figure 3 to illustrate *path crossover*.

Tree crossover consists of exchanging subtrees between two trees which similarly works with *path crossover*.

3.2 Mutation

Mutations in GAs aim to create diversity that may not have been present in the initial population. We propose a mutation called *path mutation* which extracts

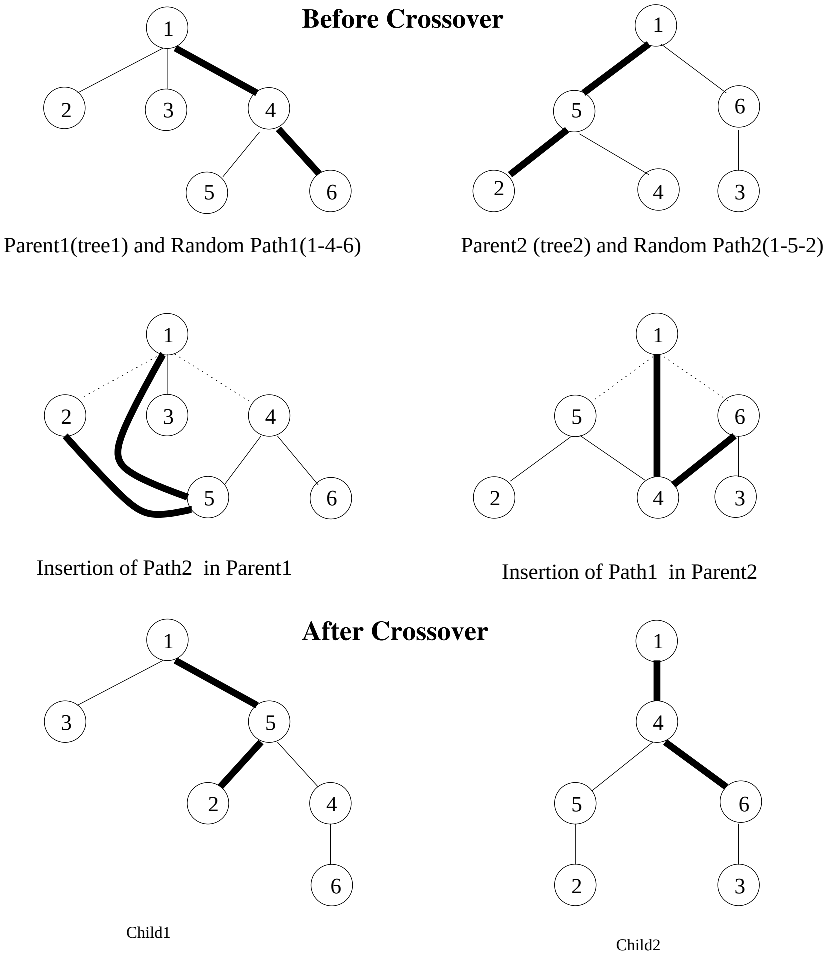

Fig. 3. Path crossover

random paths from the complete graph, and insert them in a child obtained after crossover. The procedure of *path mutation* is described in Figure 4, and an example of *path mutation* is shown in Figure 5.

Procedure pathMutation$(child)$
Begin
```
   choose a random number k
   for i = 1 to k do
   begin
      path = randomPath(completeGraph)
      insertPath(path, child)
   end
```
End.

Fig. 4. Path mutation procedure

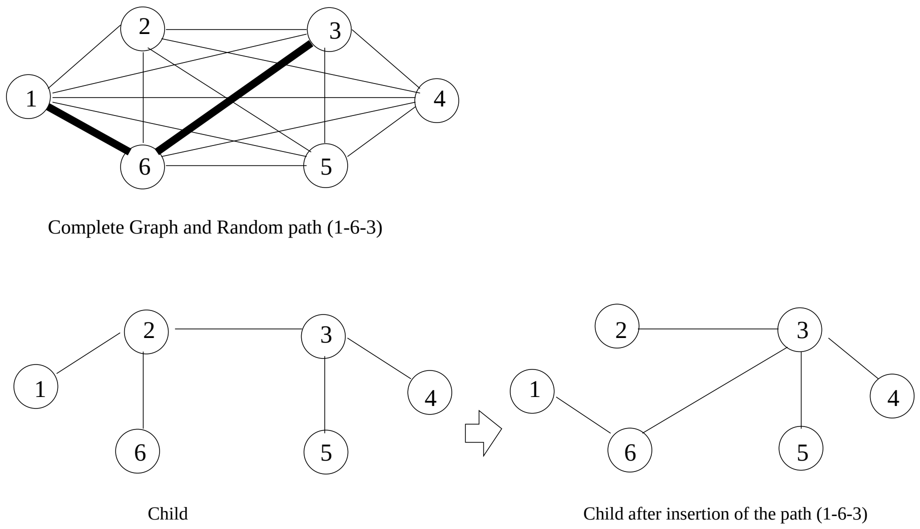

Fig. 5. Path mutation

tree mutation extracts random subtrees from the complete graph, and inserts them in a child obtained after crossover.

3.3 Complexity of crossovers and mutations

The generation of a random path or a random subtree can be effected by a modified depth-first search. The insertion of a path or a subtree in a tree is in fact the insertion of the edges of the path or the subtree. We describe the procedure of inserting an edge in Figure 6.

Procedure insertEdge($tree, e$)
Begin
```
   if e = (v1, v2) is not in tree
   then
      path = deterministPath(tree, v1, v2)
      delete an edge on path, which is not recently inserted
      add e to tree
   end
```
End.

Fig. 6. Edge insertion procedure

Therefore, the main operations of crossovers and mutations consists in searching a random or determinist path. The complexity of crossovers or mutations is then proportional to the number of searching a path multiplied by the complexity of searching a path.

4 Initial population

We propose a simple heuristic based on Prim's algorithm to randomly generate a spanning tree of a relatively small cost. We describe this heuristic in Figure 7.

Let T' be a connected subgraph of a spanning tree under construction, and span a subset of vertices $V' \subset V$. Initially V' contains some arbitrary vertex u. At each stage, the label $L(v)$, for each vertex $v \in (V - V')$, records the edge of least weight from v to V'. Each $L(v)$ is initialized to the weight $w((u,v))$ of the edge (u,v), where $d_{u,v}$ is taken as $w((u,v))$. The algorithm updates the $L(v)$ whenever a new vertex u has been added to V'.

At each iterative stage of the algorithm, a new edge e is added to T'. The edge e is the edge connecting a vertex u selected randomly from $(V - V')$ to a vertex in V' for which $w(e) = L(u)$; which is different from Prim's algorithm, where the new edge e is the edge of least weight connecting a vertex in $(V - V')$ to a vertex in V'. The algorithm stops when $V = V'$.

We illustrate this algorithm in Figure 8.

An initial population is then constructed using this heuristic from randomly generated spanning trees. It keeps random characteristics and the spanning trees in the initial population are of relatively little cost.

Procedure generateSpanningTree()
Begin

```
    randomly choose a u ∈ V
    T' ← ∅
    V' ← {u}
    for all v ∈ (V − V') do
        L(v) ← w((u, v))
    while V' ≠ V do
        begin
        randomly choose a u ∈ (V − V') and
        denote the associated edge from u to V' by e for which L(u) = w(e)
        T' ← T' ∪ {e}
        V' ← V' ∪ {u}
        for all v ∈ (V − V') do
        if L(v) > w((v, u)) then L(v) ← w((v, u))
        end
```

End.

Fig. 7. Heuristic based on Prim's algorithm to randomly generate a spanning tree

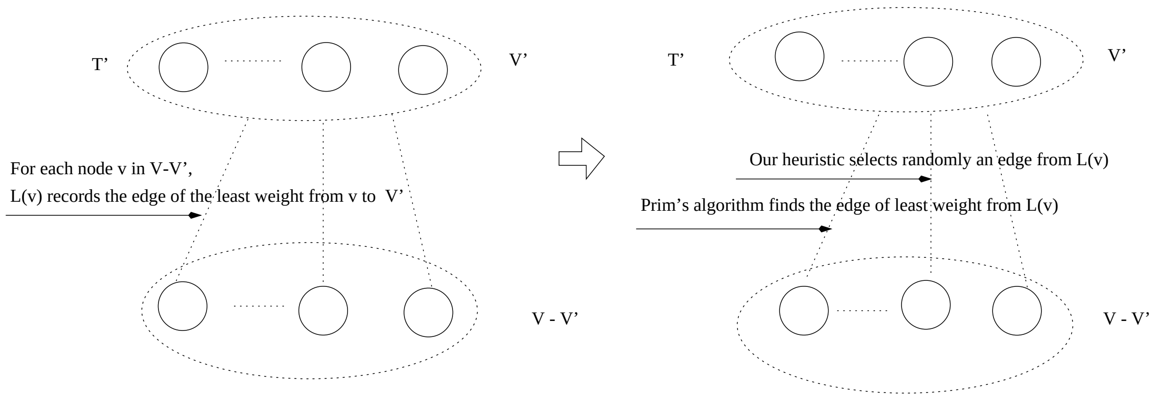

Fig. 8. Illustration of generateSpanningTree()

5 Algorithm and Results

We have implemented a GA based on the ideas discussed above and tested it on 5 examples we found in the literature.

5.1 Description of the algorithm

We give an overview of the algorithm in Figure 9.

```
Procedure geneticAlgorithm()
Begin
    intialPopulation(Φc)
    calculFitness(Φc)
    λ = bestOf(Φc)
    for gen = 1 to maxGen do
       begin
          Φn = ∅
          for j = 1 to N/2 do
             begin
                φ1 = select(Φc),φ2 = select(Φc)
                crossover(φ1, φ2, ψ1, ψ2) with pc
                mutation(ψ1), mutation(ψ1) with pm
                Φn = Φn ∪ {ψ1, ψ2}
             end
          calculFitness(Φn)
          λ = bestOf(Φn)
          Φc = Φn
       end
    output λ
End.
```

Fig. 9. Overview of our GA

One execution of the outer "for" loop corresponds to the simulation of one generation. Throught the simulation, the number of individuals $N = |\Phi_c|$ is kept constant (Φ_c is the current population). The algorithm terminates after *maxGen* generations. The procedures in the algorithm are described as follows:

- *initialPopulation* constructs an initial population Φ_c using the procedure *generateSpanningTree* described in section 4 ;
- *calculFitness* computes the fitness of each individual (spanning tree) as follows: Given a population $\Phi = \{\phi_0, \phi_1, ..., \phi_{(N-1)}\}$, let $C(\phi)$ be the cost of individual ϕ, and assume that Φ is sorted so that $C(\phi_0) \geq C(\phi_1) \geq ... \geq C(\phi_{(N-1)})$. The fitness F of ϕ_i is then computed as

$$F(\phi_i) = \frac{2i}{N-1}, \quad i = 0, 1, ..., N-1$$

This fitness computation scheme is called *ranking* (Esbensen 95);

- *bestOf* gives the individual with the highest fitness ;
- *select* selects two parents ϕ_1, ϕ_2 from Φ_c independently of each other, and each parent is selected with a probability proportional to its fitness;
- *crossover* generates two children ψ_1, ψ_2 as described in Section 3;
- *mutation* randomly changes a child as described in Section 3 with a small probability p_m.

5.2 Experimental Results

The algorithm was tested on 5 examples of the OCST problem found in the literature. These examples are 6,12,24 and 35,35 node networks represented as complete graphs, with a distance matrix and a requirement matrix.

The data of 6 and two 35 node networks are given in (Berry *et al.,* 1995, also by http://www.cse.rmit.edu.au/~rdslw/research.html), and the data of 12 and 24 node network is from the thesis of Palmer (1994), but we have not got Palmer's two other examples of 47 and 98 node networks.

Our algorithm is executed 10 times for each example. The parameters for the GA are set as below:

- the crossover probability $p_c = 0.6$;
- the mutation probability $p_m = 0.001$;
- the population size $N = 200$;
- the maximum generation $maxGen = 100$;
- $pathCrossover$, $pathMutation$.

We run our GA on 5 examples and compare to the results with those produced by Berry's GA and Palmer's GA in Table 1.

For the 6 node network, we have the same solution 534 as Berry. For the 35 node network with uniform link distance, the best solution found by Berry is 16915 and the best solution by our GA is 16420; for another 35 node network with non-uniform link distance, the best solution of Berry is 30467 obtained with 250 generations in 50 runs, and by our GA approach the optimal solution 16915 can be reached with 100 generations in 10 runs.

For 12 and 24 node networks, as we could not get the parameters used in Palmer's GA, we compare only on the result quality. We have the same solution 6.857 x 10^6 as Palmer for the 12 node network, and a better solution 2.173 x 10^6 for the 24 node network than 2.183 x 10^6 of Palmer.

We give the result of 10 consecutive runs for the instance of 35 node with non-uniform distance in Table 2. From this table, we see that our GA generates initial solutions (about 20000) near the optimal solution (16915) (while a completely random initial solution is about 200000), and then the crossovers and mutations allows to get the optimal solution.

Table 1. Comparison of solution quality for GAs

n	GA of Berry GA of Palmer	our GA
6	534	534* (after 1 generation)
35 (non-uniform distance)	30467	16915 * (after 31 generations)
35 (uniform distance)	16915	16420 (after 39 generations)
12	6.857 x 10^6	6.857 x 10^6 (after 13 generations)
24	2.183 x 10^6	2.173 x 10^6 (after 14 generations)

Table 2. Ten consecutive runs for the 35 node network with non-uniform distance

	1	2	3	4	5	6	7	8	9	10
initial solution	22157	25867	25529	24033	24297	23647	24507	21017	24887	26323
final solution	16915*	17029	18771	16915*	17653	17057	17181	17625	17181	17181

An entry marked by a * corresponding an optimal solution.

6 Conclusion and future work

In this paper, we present a new GA to solve the OCST problem in which tree structure itself is taken as chromosome, crossovers and mutations directly manipulate on trees, and an initial population is generated randomly by a simple heuristic based on Prim's algorithm. The computational results show that the proposed GA is simple and effective to get the optimal or near-optimal solution to the OCST problem.

Our work is an effort to overcome the limitation of traditional linear chromosome by allowing GAs directly work on the natural structure of problems under consideration. We also try to improve traditional uniform initialization procedure by generating randomly an initial population of "good quality".

In the future, we will test our GA on more examples and improve the implementation of crossovers and mutations. We will also adapt our GA to other problems.

Acknowledgments

We thank Berry, Murtagh, Sugden and McMahon for creating a pageweb on the OCST problem from which we get their examples. We thank Palmer for providing us his work and his some examples in his thesis. We thank also anonymous referees for their comments which helped improve this paper.

References

Abuali F.N., Wainwright R.L., Schoenefeld D.A. 1995: Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning trees problem. Proc. 6th Int. Conf. on Genetic Algorithms (ICGA'95), University of Pittsburgh, USA, 470-477

Berry L., Mutagh B., Sugden S., McMahon G. 1995: Application of genetic-based algorithm for optimal design of tree-structured communication networks. In Proceedings of the Regional Teletraffic Engineering Conference of the International Teletraffic Congress, South Africa, 361-370.

Crecenzi P., Kann V. 1998: A compendium of NP optimization problems. Available online at http://www.nada.kth.se/theory/compendium/, Aug. 1998

Esbensen H. 1995: Computing near-optimal solutions to the steiner problem in a graph using a genetic algorithm. Networks, vol. 26 (1995) 173-185.

Gibbons A. 1985: Algorithmic graph theory. Cambridge University Press, New York.

Goldberg D. E. 1989: Genetic algorithms in search, optimization and machine learning. Addison-Wesley (Reading, Mass).

Hu T. C. 1974: Optimum communication spanning trees. SIAM J. on Computing, 188-195.

Johnson D.S., Lenstra J.K., Rinnooy Kan A.H.G. 1978: The complexity of the network design problem. Networks, vol. 8(1978) 279-285

Julstrom B.A. 1993: A genetic algorithm for the rectilinear steiner problem. Proc. 15th Int. Conf. on Genetic Algorithms, University of Illinois at Urbana-Champaign (S.Forrest, Ed.). Morgan Kaufmann, San Mateo, CA, (1993) 474-479.

Kallel L., Schoenauer M. 1997: Alternative random initialization in genetic algorithms. in Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann, 268-275.

Palmer C. C. 1994: An approach to a problem in network design using genetic algorithms. PhD Thesis, Polytechnic University, Computer Science Department, Brookly, NewYork.

Palmer C. C., Kershenbaum A. 1995: An approach to a problem in network design using genetic algorithms. Networks, vol. 26 (1995) 151-163.

Peleg D., Reshef E. Deterministic polylog approximation for minimum communication spanning trees. Proc. 25th Int. Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 1443, Springer-Verlag, 670-679.

Radcliffe N.J., Surry P. D. 1995: Fundamental limitation on search algorithms: Evolutionary computing in perspective. In "Computing Science Today: Recent Trends and Developments", Lecture Notes in Computer Science, Volume 1000, page 275-291, (Ed: Jan van Leeuwen, Springer-Verlag).

Sinclair M.C. 1998: Minimum cost routing and wave-length allocation using a genetic-algorithm/heuristic hybrid approach. Proc. 6th IEE Conf. on Telecommunications, Edinburgh, UK, 66-71.

Surry P. D., Radcliffe N.J. 1996: Inoculation to initialise evolutionary search. In "Evolutionary Computing: AISB Workshop", (Ed: T. Fogarty, Springer-Verlag).

Wolpert D. H., Macready W. G. 1995: No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.