Learning the Neighborhood with the Linkage Tree Genetic Algorithm

Dirk Thierens[1] Peter Bosman[2]

[1]Universiteit Utrecht
The Netherlands

[2]CWI Amsterdam
The Netherlands

Motivation

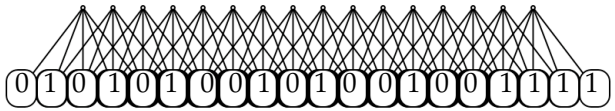
- Local search - or neighborhood search - algorithms use a predetermined, fixed neighborhood structure.
- Argument here: learning the neighborhood during search is potentially very beneficial.
- Layout
  1. Nearest-neighbor NK-landscape problem.
  2. Iterated local search.
  3. Linkage Tree genetic algorithm.

Nearest-neighbor NK-landscape

- Additively decomposable function: $\{0,1\}^\ell \rightarrow \Re$
- Overlapping, neighboring random subfunctions

$$f_{\text{NK-S1}}(\vec{x}) = \sum_{i=0}^{l-k} f_{\text{NK}}^{\text{sub}}\left(\vec{x}_{(i,\ldots,i+k-1)}\right) \quad \text{with } f_{\text{NK}}^{\text{sub}}\left(\vec{x}_{(i,\ldots,i+k-1)}\right) \in [0..1]$$

- eg. 16 subsfcts, length $k = 5$, overlap $o = 4 \Rightarrow$ stringlength $\ell = 20$



- Global optimum computed by dynamic programming
- Benchmark function: structural information is not known !
- $\Rightarrow$ Randomly shuffled variable indices.
- Experiments here: 96 subftcs, $k = 5$, $o = 4$, $\ell = 100$.

Iterated Local Search

- ILS: simple, yet powerful metaheuristic search algorithm.
  1. Local Search using a fixed neighborhood structure.
  2. Perturbation method to escape from local optima.
- Ideally, perturbation generates solutions in an adjacent region of attraction.
- ILS performs a greedy, stochastic local search in the space of local optima.

Iterated Local Search Algorithm

```
ILS()
    Sol ← LocalSearch(RandSol)
    while NotTerminated
      NewSol ← LocalSearch(Perturbate(Sol))
      if Improvement(NewSol,Sol)
        then Sol ← NewSol
    return Sol
```

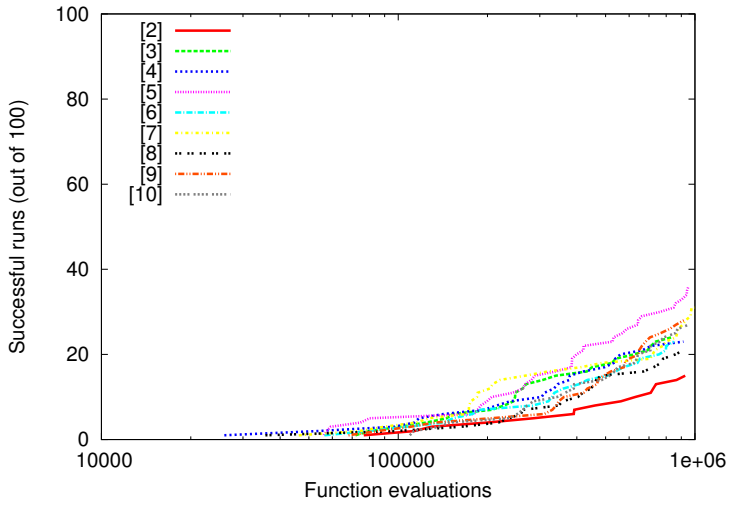



Figure: ILS with perturbation size varying from 2 to 10 bits.

Linkage Learning

- Estimation of Distribution Algorithms and Linkage Learning GAs learn a model of good solutions to match the structure of the optimization problem
- Often this modeling consists of identifying groups of problem variables that together make an important contribution to the quality of solutions.
- In general we call this dependency structure a family of subsets (FOS).
- S is the set of all problem variable indices: 0, 1, ..., L-1.
- A FOS F is a set of subsets of a main set S (F is subset of the powerset of S).
- All variable indices are element of at least one subset of the FOS F.

Linkage Tree

- Linkage Tree structure: subsets of FOS F form a hierarchical clustering.
- F = {{0,1,2,3,4,5,6,7,8,9}, {0,1,2,3,4,5}, {6,7,8,9}, {0,1,2}, {3,4,5}, {7,8,9}, {0,1}, {4,5}, {8,9}, {0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}}
- Each subset (of length > 1) is split in two mutually exclusive subsets.
- Problem variables in subset are considered to be dependent on each other but become independent in a child subset.
- For a problem of length ℓ the linkage tree has ℓ leaf nodes (the clusters having a single problem variable) and $\ell - 1$ internal nodes.

Linkage Tree Learning

- Start from univariate structure.
- Build linkage tree using bottom-up hierarchical clustering algorithm.
- Similarity measure:
  1. Between individual variables X and Y: mutual information $I(X, Y)$.
  2. Between cluster groups X_{Fi} and X_{Fj}: average pairwise linkage clustering (= unweighted pair group method with a arithmetic mean: UPGMA).

$$I^{UPGMA}(X_{Fi}, X_{Fj}) = \frac{1}{|X_{Fi}||X_{Fj}|} \sum_{X \in X_{Fi}} \sum_{Y \in X_{Fj}} I(X, Y).$$

Linkage Tree Learning

- This agglomerative hierarchical clustering algorithm is computationally efficient.
- Only the mutual information between pairs of variables needs to be computed which is a $O(\ell^2)$ operation.
- The bottom-up hierarchical clustering can also be done in $O(\ell^2)$ computation by using the *reciprocal nearest neighbor chain* algorithm.

Linkage Tree Genetic Algorithm

- The LTGA represents the dependence information in a hierarchical cluster tree of the problem variables.
- Each generation is new linkage tree is build.
- For each solution in the population the tree is traversed starting at the top.
- The nodes (= clusters) in the linkage tree are used as crossover masks.
- A random solution is selected from the population, and its values at the crossover mask replace the variable values from the current solution.
- Evaluate new solution and accept if better, otherwise reject.

Linkage Tree Genetic Algorithm

```
LTGA()
    Pop ← InitPopulation
    while NotTerminated
      Tree ← BuildTree
      forall Sol ∈ Pop:
        forall Clus ∈ Tree:
          Sol ← GreedyRecombine(Sol,Clus,Pop)
    return Sol

GreedyRecombine(Sol,Clus,Pop)
    Donor ← Random(Pop)
    NewSol ← ReplaceClusValues(Sol,Clus,Donor)
      if Improvement(NewSol,Sol)
        then Sol ← NewSol
    return Sol
```

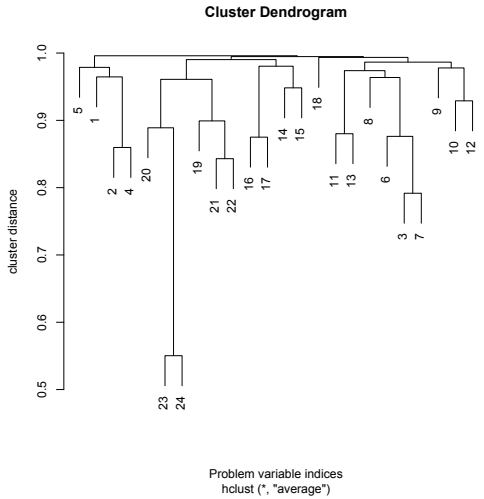


Figure: Example linkage tree for NK problem ($\ell = 24, k = 5, m = 20$).

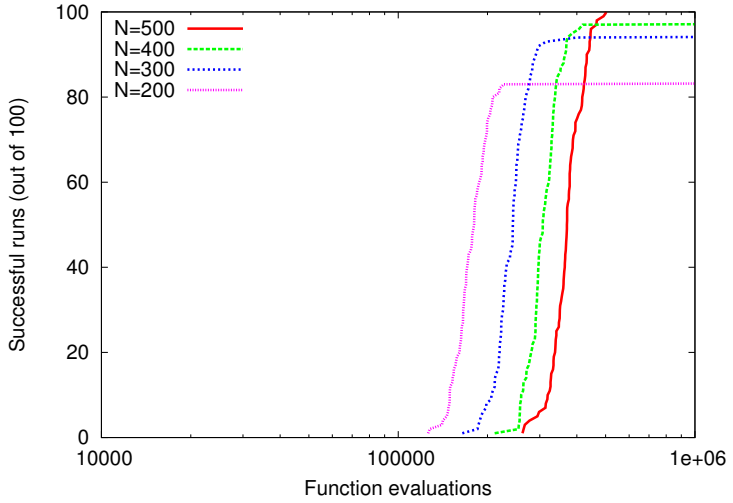


Figure: LTGA with populations sizes varying from 200 to 500.

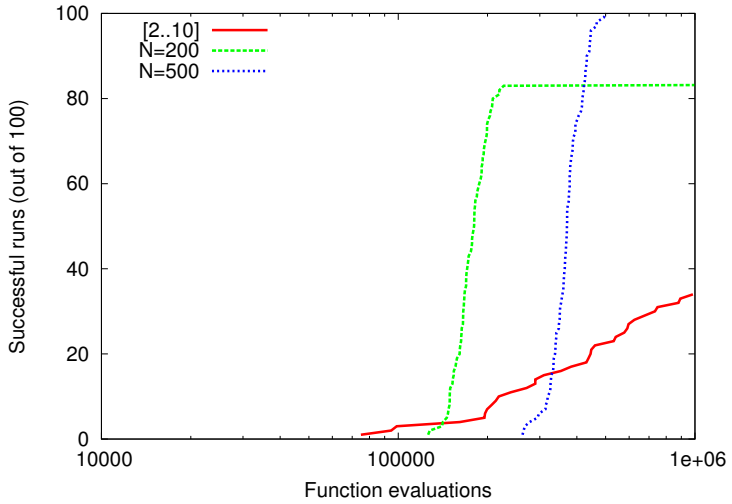


Figure: LTGA versus ILS on 100 different NK-problems.

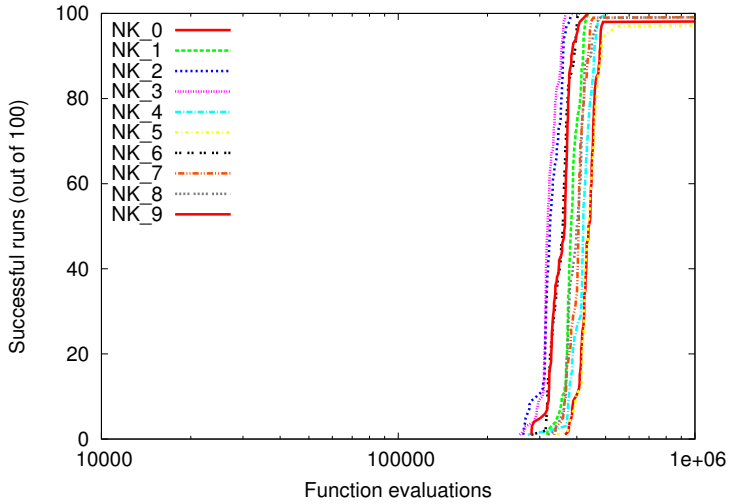


Figure: LTGA on 10 different NK-problems

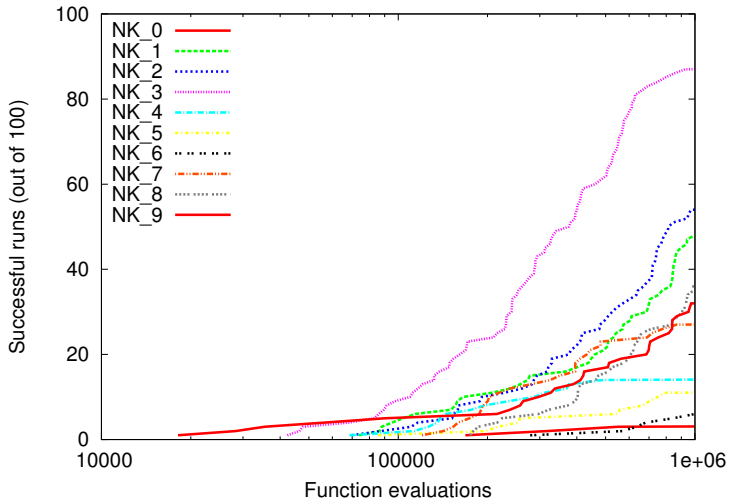


Figure: ILS with perturbation size $\in$ [2..10] for 10 NK-problems.

Conclusion

- LTGA combines structure learning with incremental, greedy recombination.
- LTGA can be viewed as a population-based, stochastic local search algorithm that each generation learns the neighborhood structure to explore.
- LTGA is much more efficient on NK-landscape problems than ILS.
- Learning the neighborhood during search can be very beneficial.
- The learned model does not need to be a perfect, structural fit of the problem.