Metaheuristic Search for Combinatorial Optimization

Dirk Thierens

Universiteit Utrecht
The Netherlands

Outline

- Combinatorial optimization problems
- Multi-start local search
- Iterated local search
- Genetic local search
- Probabilistic Model Building local search

Combinatorial optimization

Definition

A combinatorial optimization problem is specified by a finite set of solutions S and a cost function f that assigns a numerical value to each solution: $f : S \rightarrow \Re$.

- Graph Coloring
- Graph Partitioning
- Knapsack Problem
- Quadratic Assignment Problem
- Bin Packing
- Vehicle Routing
- Personnel Scheduling
- ...

Graph coloring

Definition

Given a graph $G = \{V, E\}$ where $V = \{v_1, ..., v_n\}$ is the set of vertices and $E = \{(v_i, v_j)\}$ $(i \neq j)$ is the set of edges connecting some vertices of the graph. The goal of graph k-coloring is to assign one of k colors to each vertex such that no connected vertices have the same color.

Graph Bipartitioning

Definition

Assume an undirected graph with the set of vertices V and set of edges E. The number of vertices $|V| = n$ is even. The graph bipartitioning problem is to find a partitioning of the set of vertices V into 2 subsets A and B of equal size ($|A| = |B|$), such that the number of edges between vertices of A and B is minimal.

Knapsack Problem

Definition

The knapsack problem consists of a knapsack K with fixed capacity c, and n items that have a weight w_i and profit p_i. The goal is to maximize the sum of the profits of all selected items under the constraint that the sum of their weights does not exceed the knapsack capacity.

Multi-dimensional knapsack problem: the weight of item i is given by a D-dimensional vector $\overline{w_i} = (w_{i1}, \ldots, w_{iD})$ and the knapsack has a D-dimensional capacity vector $(c_1, \ldots, c_D)$. Need to maximize the sum of the values of the items in the knapsack so that the sum of weights in each dimension d does not exceed W_d.

Local Search

aka. Neighborhood Search

- Local search algorithms iteratively try to improve the current solution by applying small changes. These changes are made by search operators.
- Only a limited set of solutions can be reached from the current solution : the neighborhood set.
- Local search explores the neighborhood of the current solution and if a better solution is found it will become the new current solution. The search continues by exploring the neighborhood of the new solution.
- Local search terminates when no improvement is found in the neighborhood of the current solution.

Local Search

Best versus First Improvement

- Best Improvement local search: searches entire neighborhood and selects the best solution if this is an improvement.
- First Improvement local search: selects the first solution encountered that is an improvement.

Multi-start Local Search (MLS)

- Local search stops when a local optimum is found
- Multi-Start local search simply restarts local search from a random initial solution
- MLS is basically doing a random search in the space of local optima
- Metaheuristics aim to improve upon performance of MLS

Local Search for Graph Coloring

Vertex Descent

1. Fix the number of color classes k.
2. For a given solution S, vertex descent iterates over all vertices in a random order .
3. For each vertex v_i all $k - 1$ vertex moves are tried. The vertex move of v_i which results in the lowest number of conflicting edges is applied to S (unless no improvement over the current solution is found; ties are broken at random).
4. When all vertices have been investigated, go back to step 2 unless the last iteration over all vertices has not resulted in a lower number of conflicting edges.

Local Search for Graph-bipartitioning

Swap neighborhood

1. Given a partition (A, B) of the node set V into two subsets A and B of equal size.
2. The swap neighborhood of (A, B) is the set of partitions (A', B') obtained by swapping a node of A with a node of B.
3. Each partition has $\frac{n^2}{4}$ neighboring partitions ($|V| = n$).

Local Search for Graph-bipartitioning

Fiduccia-Mattheyses (FM)

1. Start from a partitioning (A, B) of the graph (V, E)
2. Compute for each vertex v the gain W_v obtained by moving the vertex to the other subset
3. Create 2 arrays A and B with boundaries [- MaxDegree, + MaxDegree]. Array A (resp. B) stores at position i a list of all vertices in subset A (resp. B) with gain $W_v = i$.
4. Both arrays have an associated pointer that keeps track of the index with maximal value k
5. Initially all vertices of the graph are marked *free*.

6 If $|A| > |B|$ (resp. $|A| < |B|$) then move the vertex v from A (resp. B) that has the highest gain W_v to the subset B (resp. A). Mark the vertex v *fixed*. Fixed vertices are removed from the arrays A and B. Update the positions in the arrays A and B of the free nodes that are connected to the moved vertex.

7 Continue moving vertices until there are no free nodes left. The resulting partitioning is the same as the one we started with.

8 FM keeps track of all valid partitionings during the search process and returns the one with the lowest cut size.

9 Repeat the FM procedure until no further improvement is found.

Local Search for Knapsack Problem

Combined neighborhood

Solutions are represented by a binary vector $X^s \in \{0,1\}^n$. Random initial solutions can violate the capacity constraint: make them feasible by randomly removing items until the knapsack is filled below its capacity.

1. All items are considered in a random order and added to the current solution if they do not make the solution unfeasible.
2. The solution obtained after step 1 is further improved by considering all possible swaps between items that are in the current solution and those that are not. Whenever an item can be replaced by another item such that the fitness increases but the capacity constraint does not become violated the swap is performed.

Non-blind vs. blind knapsack problem

- In the non-blind knapsack problem algorithms can use the information of the weight and profit of individual items. A well-known fast and efficient greedy heuristic is to:
  - Sort all items in descending order of their profit/weight ratio
  - Add items in this order if their addition does not violate the capacity constraint
- Benchmark knapsack problem: generate the weights and profits of 500 items uniformly at random from the interval [10...50].
- The capacity is half the sum of the weights.

- Fitness values found with the greedy heuristic and the greedy heuristic + local search:

Knapsack problem	Greedy solution	Greedy + local search
[10:50]	10544	10545

- Fitness values found with multi-start local search after 1000 LS restarts:

Knapsack problem	Minimum fitness	25 perc. fitness	Median fitness	75 perc. fitness	Maximum fitness
[10:50]	10099	10142	10161	10181	10218

A lot of room for improvement ! $\Rightarrow$ Metaheuristic search

Outline

- Combinatorial optimization problems
- Multi-start local search
- **Iterated local search**
- Genetic local search
- Probabilistic Model Building local search

Metaheuristic Search

- Metaheuristics are search methods that aim to enhance the performance of multi-start local search by applying a problem independent strategy
- For many combinatorial optimization problems, metaheuristic search algorithms are among the best performing techniques
- Each metaheuristic specifies its own problem independent strategy.
- To be successful, the problem independent strategy of the metaheuristic (its search bias) has to coincide with the structure of the problem instance.

Iterated Local Search (ILS)

- The search strategy of iterated local search consists of applying small perturbations on local optima and restarting local search from the perturbed solution.
- Ideally the ILS perturbation step should move the search just outside the basin of attraction of the current local optimum
- If the new local optimum is better than the old one, ILS will continue searching from the new solution, otherwise it will return to the previous local optimum.
- ILS will be most successful in search space structures where the neighboring local optima have highly correlated fitness values.
- ILS is in fact performing a stochastic greedy search in the space of local optima.

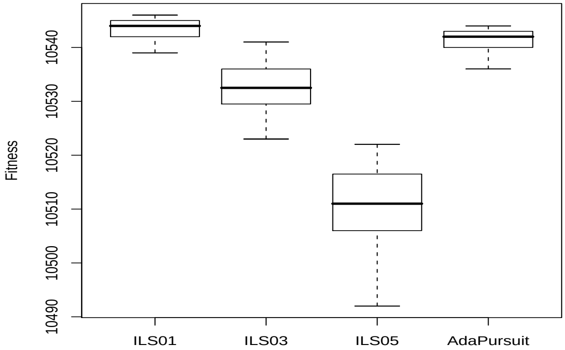

Figure: Experimental results for 20 independent runs with 1000 calls to the local search operator. Best fitness values for ILS with perturbation probability $P_{mut} = 0.01, 0.03, 0.05$ and the same 3 search operators simultaneously applied in the adaptive pursuit allocation algorithm (data set [10:50]).

Adaptive ILS

- ILS is sensitive to the choice of the perturbation step size.
- In general it is impossible to know the optimal value without experimenting with different values.
- Adaptive algorithms try to learn good values during the search.
- Example: Adaptive Pursuit strategy chooses between a set of k search operators with varying probability.

Outline

- Combinatorial optimization problems
- Multi-start local search
- Iterated local search
- **Genetic local search**
- Probabilistic Model Building local search

Genetic Local Search (GLS)

- GLS maintains a fixed size set of the best local optima encountered so far.
- New starting solutions for the local search operator are generated by recombining two local optima from the population.
- GLS will be most successful in search space structures where local optima have important partial solutions in common - and are thus shielded from destruction by a crossover operator - or have different partial solutions that can be juxtaposed to form important larger partial solutions.

GLS for the Knapsack problem

- Parent pair selected at random
- Create single offspring by uniform crossover + local search
- No duplicate solutions allowed in the population
- Offspring competes with worst solution in population

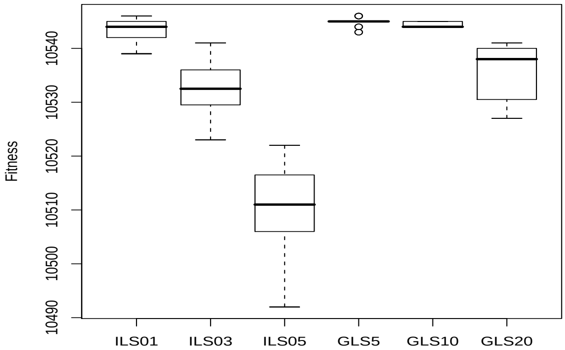

Figure: Experimental results for 20 independent runs with 1000 calls to the local search operator. Best fitness values obtained with ILS with perturbation probability $P_{mut} = 0.01, 0.03, 0.05$ and GLS with population size $n = 5, 10, 20$ (data set [10:50]).

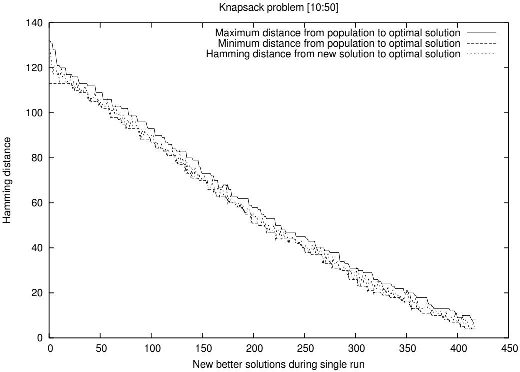

Figure: Maximum and minimum Hamming distance between the population (size $n = 20$) and the optimal solution, and between the improving solutions and the optimal solution for a single GLS run

- Uniform crossover protects the items with high profit/weight ratio and does not consider the items with low profit/weight ratio
- UX implicitly transforms the blind knapsack problem into a quasi non-blind knapsack problem
- Crossover can have multiple search biases:
  1. Random sampling within a specific subspace
  2. Juxtaposing partial solutions from two parent solutions (example: graph coloring)

Crossover for graph coloring

Two different representations:

1. Assignment representation
   - Configuration: assignment of colors to vertices

$$s : V \to \{1, ..., k\}$$

   - Basic information unit: pair vertex-color
   - Crossover: assignment crossover

$$s(v) = s_1(v) \text{ or } s_2(v)$$

Crossover for graph coloring

Two different representations:

1. Assignment representation
   - Configuration: assignment of colors to vertices

$$s : V \to \{1, ..., k\}$$

   - Basic information unit: pair vertex-color
   - Crossover: assignment crossover

$$s(v) = s_1(v) \text{ or } s_2(v)$$

2. Partition representation
   - Configuration: partition of vertices

$$s = \{V_1, ..., V_k\}$$

   - Basic information unit: subset of vertices

$$V_i = \{v_1, ..., v_n\}$$

   - Crossover: partition crossover

Partition crossover

1. Build a partial configuration of maximum size from subclasses of the color classes of the two parents
2. Complete the partial solution to obtain a full configuration

Given two parents $s_1 = \{V_1^1, ..., V_k^1\}$ and $s_2 = \{V_1^2, ..., V_k^2\}$, the partial configuration is a set $\{V_1, ..., V_k\}$ of disjoint sets of vertices having the following properties:

- each subset V_i is included in a class of one of the two parents, so all V_i are independent sets
- the union of the V_i has a maximal size
- about half of the V_i is imposed to be included in a class of parent 1 and the other half in a class of parent 2
  $\Rightarrow$ the influence of the two parents is equilibrated

The Greedy Partitioning Crossover: GPX

- Input: parent solutions $s_1 = \{V_1^1, ..., V_k^1\}$ and $s_2 = \{V_1^2, ..., V_k^2\}$
- Output: $s = \{V_1, ..., V_k\}$
- **do for** $\ell (1 \leq \ell \leq k)$
  * if ℓ is odd, then $A := 1$, else $A := 2$
  * choose i such that V_i^A has maximum cardinality
  * $V_\ell := V_i^A$
  * remove the vertices of V_ℓ from s_1 and s_2

  Randomly assign the remaining vertices of $V - (V_1 + ... + V_k)$

Example GPX

- parent $s_1 = \{(ABC), (DEFG), (HIJ)\}$
- parent $s_2 = \{(CDEG), (AFI), (BHJ)\}$
- offspring $s = \{\}$

Choose largest color class from s_1:

- parent $s_1 = \{(ABC), (HIJ)\}$
- parent $s_2 = \{(C), (AI), (BHJ)\}$
- offspring $s = \{(DEFG)\}$

Choose largest color class from s_2:

- parent $s_1 = \{(AC), (I)\}$
- parent $s_2 = \{(C), (AI)\}$
- offspring $s = \{(DEFG), (BHJ)\}$

Choose largest color class from s_1:

- parent $s_1 = \{(I)\}$
- parent $s_2 = \{(I)\}$
- offspring $s = \{(DEFG), (BHJ), (AC)\}$

Example GPX cont'd

Randomly assign vertex I:

- parent $s_1 = \{\}$
- parent $s_2 = \{\}$
- offspring $s = \{(DEFG), (BHJI), (AC)\}$

Experimental results

- Benchmark results[1] on a set of difficult and large DIMACS graphs
- Crossover for graph coloring: partitioning crossover much better suited than assignment crossover
- Hybrid GA: GPX crossover + vertex descent local search gives excellent results
- Able to find the best known solutions for most graphs in the DIMACS benchmark
- Able to find new best solutions for some largest graphs in the benchmark

[1] Celia A. Glass and Adam Prügel-Bennett. (2003). Genetic Algorithm for Graph Coloring: Exploration of Galinier and Hao's Algorithm. *Journal of Combinatorial Optimization*.

Outline

- Combinatorial optimization problems
- Multi-start local search
- Iterated local search
- Genetic local search
- **Probabilistic Model Building local search**

PMBGAs: principles

1. Probability distributions model dependencies between problem variables present in good solutions
2. Selection makes these fitness-based dependencies stand out
3. Estimating a probability model over the selected solutions identifies these dependencies
4. Drawing new samples from the probability model will respect the dependencies

PMBLS

- Probabilistic model-building GA + local search = PMBLS
- Bivariate probabilistic model: learns the pairwise dependencies between problem variables
- Building a dependency tree = maximum spanning tree over the dependency graph
- New solutions obtained by sampling from the dependency tree
- Bivariate PMBLS for Graph Bipartitioning

Bivariate Probabilistic Model for Graph Bipartitioning

1. Redundancy problem
   - Redundancy problem easy to solve for crossover
   - Probabilistic model: count frequencies that two vertices are in same partition
   - (00 or 11) versus (01 or 10)

Bivariate Probabilistic Model for Graph Bipartitioning

1. Redundancy problem
   - Redundancy problem easy to solve for crossover
   - Probabilistic model: count frequencies that two vertices are in same partition
   - (00 or 11) versus (01 or 10)
2. Dependency value
   - Dependency tree build over most extreme frequency values
   - Low values as important as high values
   - Build dependency tree over max(p, 1-p) values

Bivariate Probabilistic Model for Graph Bipartitioning

1. Redundancy problem
   - Redundancy problem easy to solve for crossover
   - Probabilistic model: count frequencies that two vertices are in same partition
   - (00 or 11) versus (01 or 10)
2. Dependency value
   - Dependency tree build over most extreme frequency values
   - Low values as important as high values
   - Build dependency tree over max(p, 1-p) values
3. Computational complexity
   - Standard Bivariate PMBGA computes pairwise frequencies between all variables: $O(|V|^2)$
   - Computational complexity would be larger than the complexity of the FM local search !
   - Solution: reduce computational complexity by only considering the pairwise interactions between connected vertices

Experiments

All solutions obtained by applying FM local search algorithm to the initial and offspring solution.

- MLS: generate 1000 local optima with FM algorithm from random initial solution
- GLS: steady-state GA population size 50, parents randomly selected, uniform crossover, offspring competes with the worst solution in the population.
- PMBLS: population size 100, dependency tree constructed from 50 best solutions, 50 new samples added.

Benchmark graphs

- Widely used benchmark problems: U500.d and G500.p graphs
- U500.d graphs
  - 500 vertices: randomly chosen within the unit square
  - vertices within distance $\sqrt{\frac{d}{500\pi}}$ are connected
  - expected vertex degree = d
  - $d = 0.05, 0.10, 0.20, 0.40$
- G500.p graphs
  - 500 vertices: with probability p connection between any pair
  - expected vertex degree = $p(500 - 1)$
  - $p = 0.005, 0.01, 0.02, 0.04$

Performance

Fixed number of local optima

Table: multi-start local search, genetic local search, and the bi-variate probabilistic model-building GA: each generating 1000 local optima - this is, 1000 calls of FM.

	MLS		GLS		PMBLS	
	mean	std	mean	std	mean	std
U500.05	8.62	1.90	4.24	1.35	3.62	0.98
U500.10	26.06	0.42	26.02	0.14	26	0
U500.20	178	0	178	0	178	0
U500.40	412	0	412	0	412	0
G500.005	53.28	0.85	51.3	0.46	51.48	0.57
G500.01	223.52	1.66	218.36	0.84	219.2	1.48
G500.02	630.64	1.48	627.68	1.05	627.94	1.17
G500.04	1749.98	2.45	1745.54	1.51	1746.5	1.85

Table: two-tail p-values for the unpaired t-test: values smaller than 0.05 indicate a statistical significant difference between the mean values of the best local optima found.

	MLS/GLS	MLS/PMBLS	GLS/PMBLS
U500.05	$<$ **0.001**	$<$ **0.001**	**0.010**
U500.10	0.524	0.315	0.315
U500.20	-	-	-
U500.40	-	-	-
G500.005	$<$ **0.001**	$<$ **0.001**	0.085
G500.01	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**
G500.02	$<$ **0.001**	$<$ **0.001**	0.245
G500.04	$<$ **0.001**	$<$ **0.001**	**0.005**

Efficiency

Fixed run time

Table: multi-start local search, genetic local search, and the bi-variate probabilistic model-building GA: run time for 1000 optima with PMBLS ≈ 1250 optima with MLS ≈ 1500 optima with GLS.

	MLS		GLS		PMBLS	
	mean	std	mean	std	mean	std
U500.05	8.54	1.71	3.48	1.39	3.62	0.98
U500.10	26	0	26	0	26	0
U500.20	178	0	178	0	178	0
U500.40	412	0	412	0	412	0
G500.005	53.5	0.83	51.14	0.35	51.48	0.57
G500.01	223.48	1.71	218.4	1.11	219.2	1.48
G500.02	630.62	1.77	627.46	0.94	627.94	1.17
G500.04	1749.38	2.49	1745.34	1.35	1746.5	1.85

Table: two-tail p-values for the unpaired t-test: values smaller than 0.05 indicate a statistical significant difference between the mean values of the best local optima found.

	MLS/GLS	MLS/PMBLS	GLS/PMBLS
U500.05	$<$ **0.001**	$<$ **0.001**	0.562
U500.10	-	-	-
U500.20	-	-	-
U500.40	-	-	-
G500.005	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**
G500.01	$<$ **0.001**	$<$ **0.001**	**0.003**
G500.02	$<$ **0.001**	$<$ **0.001**	**0.026**
G500.04	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**

Observations

- Geometric graphs U500.10, U500.20, and U500.40 are easy enough for MLS to find the optimal solution
- U500.05 graph: MLS outperformed by GLS and PMBLS; PMBLS outperforms GLS for a fixed number of calls to FM, but this difference disappears when GLS can explore 50% more local optima (= same run time)
- Random graphs G500.p: MLS always outperformed by GLS and PMBLS
- Random graphs G500.p: GLS slightly more efficient than PMBLS

Discussion

- Both metaheuristics (GLS and PMBLS) have better performance and efficiency than MLS
- PMBLS has a better performance than GLS for the difficult geometric graph U500.05, however efficiency-wise there is no difference
- For the random graph problems (G500.p) the efficiency gain of GLS makes it the preferred technique

Conclusion

- Local search is a powerful paradigm to solve large scale combinatorial problems
- Multi-Start local search is basically a random search in the space of local optima
- Metaheuristics try to improve the efficiency of MLS following a problem independent search strategy
- Practitioner's point of view:
  natural progression from MLS $\rightarrow$ ILS $\rightarrow$ GLS $\rightarrow$ PMBLS