Linear Genetic Programming of Metaheuristics

Robert E. Keller
Department of Computer Science
University of Essex, UK
rkeller@essex.ac.uk

Riccardo Poli
Department of Computer Science
University of Essex, UK
rpoli@essex.ac.uk

ABSTRACT

We suggest a flavour of linear Genetic Programming in domain-specific languages that acts as a hyperheuristic (HH).

Categories and Subject Descriptors: I.2.2 Artificial Intelligence [**Automatic Programming**]: Program synthesis

General Terms: Algorithms

Keywords: Genetic Programming; Metaheuristics, Optimization

A HH attempts building a metaheuristic (MH) that is good in the sense that it locates acceptable solutions to a given problem in feasible time. However, due to the NFL situation during optimisation, a fixed HH that efficiently operates for all domains cannot be designed. Thus, we suggest a generic HH where a grammar G describes the structure of MHs specific to a given domain D, while one can exchange G with a grammar for another domain. Therefore, a MH is a sentence $l \in L(G)$, the language of G. We realize this framework with techniques from linear GP. Thus, the GP HH (Algorithm 1) considers a MH as a genotype $g \in L(G)$. See [2] for a detailed description of algorithms and more results from the work presented here.

Given a grammar G with terminal set T, we get $g \in L(G) \subset T^*$, the set of all strings over T. Each primitive $t \in T$ stands for an operator that is a heuristic or part of one. Therefore, g represents a series of operator applications that grows a structure, s, that is a candidate solution of a given problem. We define g's fitness as the quality of s.

Initialization and mutation may result in a primitive-sequence, $\sigma \in T^*$, with $\sigma \notin L(G) \subset T^*$. In this case, we invoke a mapping function, m, to derive $\sigma' \in L(G)$. Over the population of the GP HH, m implies a variance of the effective genotype size, which is beneficial as it is a necessary condition for the emergence of parsimonious, good metaheuristics. In combination with point mutation and a fixed maximal size of genotypes, m also implicitly counters bloat.

We observe the behaviour of the HH on traveling-salesperson problems (TSP). We provide the HH with two classic, trivial, TSP-specific heuristics: `2-change` and `3-change`. We add i) `IF_2-change` that only executes the change if it shortens the tour under construction, ii) and its twin, `IF_3-change`. We also introduce `REPEAT`: given $p \in T$, $\iota \in \mathbb{N}$, it executes p until a shorter tour results or until p has been executed ι times.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
ACM 978-1-59593-697-4/07/0007.

Algorithm 1 GP HH: grammar G, p, s, μ, ω

1: create p genotypes in L(G) with fixed maximal size s
2: **while** not yet ω genotypes produced
3: *Selection*: `2-tournament`
4: *Reproduction*: `Copy` winner g into loser's place $\rightarrow g'$
5: *Exploration*: with probability μ
`Point-mutate` $g' \rightarrow h$; $\mathsf{m}(h) \rightarrow g''$

Table 1: HH performance over 100 runs.

eil51	Mean best	SD	Best	ι
HH	528.89	8.98	508.75	10
"	**428.87**	**0.00**	**428.87**	**800**
Hybrid GA	—	—	**428.87**	—
eil76	Mean best	SD	Best	ι
HH	600.09	12.37	576.60	800
"	586.29	12.81	*559.78*	15,000
Hybrid GA	—	—	**544.37**	—

We consider problem `eil51` from TSPLIB, a standard benchmark suite, with $\frac{50!}{2} \approx 1.5 \times 10^{64}$ tours. We set $\mathsf{p} = 100$, $\mathsf{s} = 500$, $\omega = 100,000$, μ=0.5, and we define a grammar that merely allows for sequences built from `2-change`, `IF_2-change`, `IF_3-change`, "`REPEAT IF_2-change`", and "`REPEAT IF_3-change`".

For $\iota = 800$, each of 100 independent HH runs produces at least one MH that finds a tour whose length equals the best known result (see Table 1, where column "Best" gives the length of the shortest cycle found over all runs for given ι). On average, a run lasts 10.1 min, with 1–2 metaheuristics being produced each 10ms, using a single core of an Intel Xeon 3.2 GHz machine.

We also consider `eil76` with about 1.2×10^{109} tours. The "Hybrid GA" rows in Table 1 give the best known tour lengths, taken from [1] that presents a solver, only applicable to Euclidean TSPs, that uses several specialized, nontrivial, handcrafted heuristics. Remarkably, on the mentioned, large solution spaces, evolved MHs match or approach the effectiveness of the specialized solver.

1. REFERENCES

[1] G. Jayalakshmi, S. Sathiamoorthy, and R. Rajaram. An hybrid genetic algorithm. *International Journal of Computational Engineering Science*, 2(2):339–355, 2001.

[2] R. E. Keller and R. Poli. Linear Genetic Programming of Metaheuristics. Technical Report CSM-468, Department of Computer Science, University of Essex, 2007.