Improved Benchmark Results from Subheuristic Search

Robert E Keller and Riccardo Poli

Department of Computing and Electronic Systems, University of Essex, United Kingdom
rkeller@essex.ac.uk, rpoli@essex.ac.uk

1 Introduction

In [3] we proposed a hyperheuristic (HH) driven by Genetic Programming (GP). In given, more or less problem-specific target languages, this GP-HH expresses its evolved metaheuristics (MH), which makes the HH a generic solver. Here, we demonstrate that, for larger search spaces, specific MHs, evolved over simple and less specific languages, may outperform man-made, specific MHs. The HH keeps working well when the size of the search space increases and less specific languages are provided that involve subheuristics, i.e., parts of heuristics. Remarkably, for larger spaces, the HH evolves MHs that improve on benchmark results set by a sophisticated, specific, man-made solver. Furthermore, we show that giving modest, specific knowledge is sufficient also when the HH deals with larger spaces.

A MH, M, built from given *primitives*, is represented as a sentence from a given language. M holds an individual repetition value, ι_M, that determines how often each call of an iterative primitive I of M may repeat, at most, a search step given to I. The GP-HH co-evolves MHs and their ι values.

2 Results

The set of NP-hard travelling salesperson problems (TSP) is an appropriate sample domain. With all n nodes of a problem given, one describes a cycle (tour) as a permutation of nodes, $p = (v_0, ..., v_{n-1})$, over $\{0, ..., n-1\}$. We call permutation $(0, 1, ..., n-1)$ the *natural* cycle of the problem. The primitive `NATURAL` creates this cycle. For a tour, the low-level heuristic `2-CHANGE`, when given two edges $(a,b),(c,d) : a \neq d, b \neq c$, replaces them with $(a,c),(b,d)$. Another primitive, `IF_2-CHANGE`, executes `2-CHANGE` only if this improves the tour under construction. Similarly, we also supply a heuristic that we call `IF_3-CHANGE`.

"`REPEAT_UNTIL_IMPROVEMENT` p" is a primitive that, as a component of a metaheuristic M, keeps executing p, a primitive, until this leads to a better result or until p has been executed ι_M times. *Complete*, a grammar over the described primitives, is shown in Figure 1. For a given tour, the probably simplest conceivable subheuristic randomly selects a node and swaps it with one of its direct neighbours, here, its right one. We call this operator `SWAP_NODE`. Figure 2 shows the grammar rule resulting from the addition of `SWAP_NODE`. We call the associated language *Swap*.

We shall only be dealing with real-valued distances, while, for historic reasons, literature also mentions integer tour lengths resulting from rounding. We consider problem `eil51` from TSPLIB. Its dimension is $n = 51$ nodes, and its best high-precision solution known has a length of 428.871765 as discovered by a MH evolved by the GP-HH [2].

```
metaheuristic       ::= NATURAL
                      | NATURAL search
search              ::= heuristic
                      | heuristic search
heuristic ::= 2-CHANGE
            | loop IF_2-CHANGE
            | loop IF_3-CHANGE
loop      ::= REPEAT_UNTIL_IMPROVEMENT
            | /* empty */
```

Fig. 1. Description of language *Complete*.

```
heuristic ::= 2-CHANGE
            | loop IF_2-CHANGE
            | loop IF_3-CHANGE
            | SWAP_NODE
```

Fig. 2. Language *Swap*. Other rules as given in Figure 1.

Table 1. Effectiveness over *Complete* and *Swap*, 100 runs each, max. iteration value $\iota_{max} = 1{,}000$. Parameters: Popul. size 100, Genotype size 500, Offspring 100,000, Mut. prob. 0.5. **P.%**: "**Mean** best" in % of best low-precision result known from literature [1], $\alpha = 428.87$. **Best**: shortest length over all runs.

eil51	Mean best	S.D.	Best	P.%
Complete	428.9	0.17	**428.872**	0.006
Swap	429.98	1.29	**428.872**	0.26

Table 2. Effectiveness over *Swap*. Parameters: Popul. size 100; Offspring 1,000,000; Mut. prob. 0.5; ι_{max} 2,000. **P.%**: " Mean best" in % of best high-precision results known from literature: $\alpha_{76} = 544.36908$ [2]; eil101: $\alpha_{101} = 640.975$ [1]. *r* runs. g: genotype size

eil	Mean best	S.D.	Best	P.%	g	*r*
76	545.38	1.001	α_{76}	0.19	900	100
101	645.12	1.2	641.697	0.65	1,800	32

This reproduces or nearly reproduces the best-known low-precision result established by a sophisticated man-made MH [1]. We give performance results in Table 1. Top-quality MHs ("Best") are easily found, and effectiveness is well within $1/3$% of the best result known. Thus, adding flexibility at the subheuristic level does not jeopardise effectiveness for smaller search spaces.

Next, we consider eil76, a 76-node problem, and eil101 (101 nodes). Parameter values and results are given in Table 2. For eil76, with genotype size $g = 900$, one sees that the best solution known and other, very good solutions ("Mean best") are found with high reliability ("S.D."). For eil101, we double the genotype size, while the search-space size increases by about factor 10^{48}. Still, our best solution is around 0.1% of α_{101}, the best high-precision result known. We conclude that, with a comparatively modest increase of resources, GP-HH effectiveness scales up favourably.

We give another, generic subheuristic, IF_improve, that precedes a problem-specific primitive, p, in which case p is executed only if this improves the current solution. Adding IF_improve to the previous language description gives language *Ifi* (Figure 3). We perform experiments over *Ifi*, applying the following parameter values: population size 100; offspring 1,000,000; mut. prob. 0.5; ι_{max} 2,000; genotype size 900. Table 3 gives results. For eil76, one sees that the best solution known and other very good solutions are found, so that the *average* best is within $1/2$% of the best solution known. Thus, the primitive IF_improve supports effectiveness of the hyperheuristic search. It

```
heuristic ::= 2-CHANGE
            | ifi SWAP_NODE
            | loop IF_2-CHANGE
            | loop IF_3-CHANGE
ifi       ::= IF_improve
            | /* empty */
```

Fig. 3. Language *Ifi*. Other rules as given in Figure 1.

Table 3. Effectiveness over *Ifi* and *NN*. Parameters: Table 2. **P.%**: *Mean* best over all **r** runs in % of best high-precision result known for eil76: $\alpha_{76} = 544.36908$, [2]; eil101: $\alpha_{101} = 640.975$, d198: $\alpha_{198} = 15,876.38$, [1]. **g**: genotype size. **Bold** value is new, best real-valued result known.

Problems	Mean	S.D.	Best	P.%	g	r
76	546.85	1.14	α_{76}	0.46	900	100
101	659.68	3.70	651.250	2.9	900	100
101_d	648.33	2.16	641.302	1.15	1,800	100
101_dNN	640.95	0.28	**640.212**	-0.004	1,800	24
$d198_d$NN	15,909.7	13.57	**15,852.177**	0.21	1,800	100

gives evolution more flexibility in balancing greedy vs randomised search. For instance, while the primitive SWAP_NODE represents a purely random search step, it may be preceded by IF_improve which yields a greedy step.

For eil101, the mean best is within 3% of the best solution known. This is remarkable because no GP-HH parameter was changed, while the search space is significantly larger. For $\texttt{eil101}_d$, i.e., eil101 approached with doubled genotype size ($g = 1,800$), the mean best is well within $4/3$% of α_{101}, the best solution known so far. Our best solution is within 0.05% of α_{101}. So, here, the GP-HH, given comparatively few additional resources, produces MHs that deal very effectively with a much larger space.

Next, we provide very modest problem knowledge in the shape of a *randomised nearest-neighbour* heuristic R that replaces NATURAL: randomly add a node *n* as first node of a permutation *p* under construction, add a non-added node *m* that is nearest to *n*, repeat adding step for *m*, and so forth, until completion of *p*. This method, while somewhat more complex than NATURAL, is trivial when compared to the sophisticated initialisation heuristics used in man-made MHs. We call the corresponding language *NN* and the related experiment $\texttt{eil101}_{d\,\text{NN}}$ (Table 3). Overall effectiveness and reliability strongly improve compared to $\texttt{eil101}_d$ (language *Ifi*). In particular, one of the best evolved MHs, $\mu_{\alpha NN}$, improves on the best known real value, α_{101}, delivered by a hand-crafted, problem-specific algorithm, by about 0.12%. We give this *new benchmark value* with high precision: $\alpha_{101NN} = 640.211609$. For d198 (TSPLIB) with a clearly larger search space, with unchanged GP-HH parameter values, effectiveness is also excellent. Most remarkably, one of the best evolved MHs improves on the best known real value, α_{198}, delivered by a man-made method, by about 0.15%. We give the *new benchmark value* with high precision: $\alpha_{198NN} = 15,852.176758$.

References

1. G. Jayalakshmi and S. Sathiamoorthy, et. al. An hybrid genetic algorithm. *International Journal of Computational Engineering Science*, 2001.
2. R. E. Keller and R. Poli. Linear genetic programming of parsimonious metaheuristics. In D. Srinivasan et al., eds., *2007 IEEE CEC*, 2007.
3. R. E. Keller and R. Poli. Toward Subheuristic Search. In *2008 IEEE WCCI*, 2008.