The probabilistic heuristic in local (PHIL) search meta-strategy

Randall, M

Published in:
Innovations in applied and artificial intelligence

DOI:
10.1007/11504894_90

Published: 01/01/2005

Document Version:
Peer reviewed version

Licence:
Unspecified

The Probabilistic Heuristic In Local (PHIL) Search Meta-strategy

Marcus Randall[1]

Faculty of Information Technology
Bond University
QLD 4229
Australia

Abstract. Local search, in either best or first admissible form, generally suffers from poor solution qualities as search cannot be continued beyond locally optimal points. Even multiple start local search strategies can suffer this problem. Meta-heuristic search algorithms, such as simulated annealing and tabu search, implement often computationally expensive optimisation strategies in which local search becomes a subordinate heuristic. To overcome this, a new form of local search is proposed. The Probabilistic Heuristic In Local (PHIL) search meta-strategy uses a recursive branching mechanism in order to overcome local optima. This strategy imposes only a small computational load over and above classical local search. A comparison between PHIL search and ant colony system on benchmark travelling salesman problem instances suggests that the new meta-strategy provides competitive performance. Extensions and improvements to the paradigm are also given.

Keywords: heuristic search, combinatorial optimisation, meta-heuristic.

1 Introduction

Local search is a classical approach to solving combinatorial optimisation problems (COPs). There have been numerous instances of local search algorithms being used by themselves to solve COPs (e.g., [3, 5, 10, 12]) (usually as a means of implementing a control strategy); as the basis of meta-heuristic search strategies (e.g., simulated annealing (SA) [14] and tabu search (TS) [8]); or as an adjunct heuristic to other heuristics/meta-heuristics (e.g., ant colony optimisation (ACO) [4], greedy randomised adaptive search procedures (GRASPs) [6]). While the iterative meta-heuristic search strategies (such as SA and TS) are able to use local search to overcome local optima (usually at the expense of long runtimes), the settling in local minima or maxima for the classical approach is a limitation. However, the cost for using meta-heuristic strategies is that they can require significant amounts of computational runtime beyond that of the local search component. The Probabilistic Heuristic In Local (PHIL) search is

designed to extend classical local search by augmenting it with a computationally inexpensive probabilistic branching strategy. This branching strategy is a recursive one that continues the search process from a point within the current search trajectory.

The remainder of the paper is organised as follows. Section 2 discusses other extensions to local search while Section 3 describes the extensions to the classic algorithm that constitute PHIL search. Section 4 presents the results of the computational experiments using benchmark travelling salesman problem (TSP) instances. Additionally, a comparison to an implementation of ant colony system (ACS) is provided. Finally Section 5 provides a discussion of some of the extensions and enhancements that are possible for the new search strategy.

2 Local Search

There have been a number of variations of local search that have been extended from the previously described classical forms. Some of the more notable approaches are described below.

Guided Local Search (GLS) [9, 15] is a nominal extension to classical local search that enables it to become a meta-strategy. Once local search becomes stuck in a local optimum, the meta-strategy component is activated. The weights/penalties in an augmented objective function are increased so as to guide the local search out of the particular local optimum. This is a form of search space transformation that has only been applied to a few combinatorial optimisation problems. An extended version of the algorithm in which tabu style aspiration criteria and random moves are added gives comparable performance on the quadratic assignment problem to standard TS approaches [9].

The Affine Shaker algorithm of Battiti and Techolli [1, 2] works by successively sampling sub-regions of search space. Each region is defined by a central starting point (i.e., the region surrounds this point equally). This region is then sampled to generate a new tentative point. Depending on whether this new point is of better or worse quality, the sampling area is expanded or compressed (respectively). If the sampling is able to produce a better solution, this becomes the new starting point, and the sub-region is relocated around this point. Thus the process can continue for a number of iterations. The affine shaker algorithm has been applied to problems within neural networking back propagation [2] and as part of continuous reactive tabu search solving benchmark functions [1].

Paquette and Stützle [10] present an enhancement of local search called Iterated Local Search (ILS) that optimises problems, such as the graph colouring problem, in which there are two optimisation criteria. In terms of this problem, ILS first attempts to find feasible colourings for successively smaller chromatic numbers. At each iteration of the algorithm, a complete local search based heuristic (such as classic hill climbing or tabu search) is executed. The procedure terminates once a legal colouring cannot be found and hence returns the chromatic number of the previous colouring. The authors reported receiving comparable results to state of the art heuristics and meta-heuristics on benchmark problems.

Yuret and de la Maza's [16] Dynamic Hill Climbing algorithm is a population based approach that uses genetic algorithm mechanisms of reproduction and selection in order to modify solutions. It also adds two elements to the search. These are: a) the dynamic alteration of the search space co-ordinate system and b) the exploitation of local optimum. The first is necessary when the search encounters a local optima. It re-orients the search space co-ordinate system in order to compute an escape trajectory. In terms of the latter, the local optima found by the search process are recorded. If the population becomes stuck, a diversification measure is enacted. A new starting point point is generated by maximising the Hamming distance between the nearest recorded local optimum. At this stage, the search process is restarted and the list of local optima is reset. Dynamic hill climbing has been applied to De Jong's set of continuous test functions and has provided competitive performance [16].

Unlike the previously described local search methods, Complete Local Search [7] implements a local search having a memory component. The strategy keeps a finite list of previously visited solutions. This list is used to prohibit the search process from exploring the neighbourhoods of these solutions at a later stage. Experimental evaluation on the travelling salesman and subset sum problem instances [7] suggest that though its execution times are efficient, its overall performance is not yet comparable with standard heuristic and meta-heuristic implementations.

3 The PHIL Search Algorithm

PHIL search is an extension of classical local search. It resembles multistart local search as it performs multiple local searches. The key difference is that instead of starting at a random point in state space, PHIL search probabilistically chooses a point within the recently completed local search trajectory. The rationale for this is that the point will at least be better than the starting point and may lead to a superior end point. At this point, the new local search (referred to as a branch) chooses the next best transition operation[1] and proceeds until no more improvement is possible (i.e., the classic local search termination condition). Moreover, this is a recursive process as once a local search trajectory has been explored (in terms of the generation of probabilistic branch points), the strategy will return to the branch from which the current branch was initiated. This is consistent with depth first search behaviour.

Termination of the overall algorithm is either after a certain number of individual PHIL searches have been executed, or when a particular solution cost has been obtained. In terms of the former, an individual PHIL search is completed once the root branch (the original local search trajectory) has explored all its branch points. These may be referred to as search trees. The only parameter required by PHIL search (referred to as α) is the probability of branching at a point on the local search trajectory. A high probability will produce dense search trees, while the reverse is true for a low probability.

[1] Any standard local search operator can be used within PHIL search.

Algorithm 1 The initialisation phase of PHIL search

```
1: Get user parameters(α, num_restarts)
2: for trial = 1 to num_restarts do
3:    x = Generate a random initial feasible solution
4:    cost = Find_cost(x)
5:    Initialise all of index array elements to 0
6:    cost = Perform_phil(x, α, cost, index, 1)
7:    if cost < best_cost then
8:       best_cost = cost
9:    end if
10: end for
11: Output best_cost
12: end
```

Algorithm 2 The PHIL search strategy

```
1: Perform_phil(x, α, cost, index, level)
2: x' = x
3: cost, trail_length = Perform_local_search(x', cost, tran_list1, tran_list2)
4: index[level] = Probabilistic_find_branch_point(x, α, tran_list1, tran_list2)
5: if index[level] ≠ dead_branch then
6:    index[level] = index[level] + 1
7:    level = level + 1
8:    Perform_phil(x, α, cost, index, level)
9:    level = level − 1
10: else
11:    return cost
12: end if
13: end Perform_phil
```

Algorithms 1-4 give the combined pseudocode description of PHIL search. The first presents the framework in which PHIL search is executed. The termination condition used here represents the number of search trees generated. The overall PHIL strategy is given in Algorithm 2 while Algorithm 3 corresponds to a standard local search procedure. The final part of PHIL search probabilistically chooses a branching point on the current local search trajectory. Fig. 1 provides an explanation of some of the terms used within the overall algorithm.

4 Computational Experience

The methodology and results of testing PHIL search are described herein. The target application for this initial study is the TSP. The local search operator is the inversion operator, as it has been shown to be effective by Randall and Abramson [12].

Initial experimentation with the α parameter suggests that appropriate values of it are a function of the size of the problem. In this case, the term "appropriate" refers to values that tend to produce good quality solutions. Using a

Algorithm 3 The local search component of PHIL search

```
1: Perform_local_search(x, cost, tran_list_1, tran_list_2)
2: new_cost = cost
3: index = 1
4: while new_cost < cost do
5:    cost = new_cost
6:    neighbours = Evaluate_neighbours(x)
7:    tran_list_1[index] = neighbours[1]
8:    if there is a second best transition then
9:       tran_list_2[index] = neighbours[2]
10:   end if
11:   Apply_transition(x, tran_list_1[index])
12:   new_cost = Find_cost(x)
13:   index = index + 1
14: end while
15: return new_cost and index
16: end Perform_local_search
```

Algorithm 4 The probabilistic branching strategy within PHIL search

```
1: Probabilistic_find_branch_point(x, trail_length, α, tran_list_1, tran_list_2, index)
2: Perform all transitions up to and including the index^th
3: while found = false And index < trail_length do
4:    Apply_transition(x, tran_list_1[index])
5:    q = unif_rand()
6:    if q ≤ α And tran_list_2[index] is present then
7:       Apply_transition(x, tran_list_2[index])
8:       return index
9:    end if
10:   index = index + 1
11: end while
12: return dead_branch
13: end Probabilistic_find_branch_point
```

linear regression model on a subset of the test problem instances revealed that $\alpha = -0.008n + 0.925$ (where n is the number of cities and the minimum bound of the equation is 0.005) is a good overall function for the TSP. The investigation of this parameter will receive further attention in future studies.

4.1 Methodology and Problem Instances

The computing platform used to perform the experiments is a 2.6GHz Red Hat Linux (Pentium 4) PC with 512MB of RAM.[2] Each problem instance is run across ten random seeds.

The experiments are used to compare the performance of PHIL search to a standard implementation of ACS (extended details of which can be found

[2] The experimental programs are coded in the C language and compiled with gcc.

Fig. 1. Terms used within the PHIL search algorithm

x is the solution vector, Find_cost evaluates the objective function, *dead_branch* signifies a branch that has been explored, Evaluate_neighbours evaluates all the neighbours of a solution using a defined local search operator, *neighbours* is an ordered array of transition attributes of the neighbours, $tran\_list_1$ refers to the list of best transitions at each stage of the local search while $tran\_list_2$ is the list of the second best, Apply_transition() applies a transition to a solution using a set of transition attributes and unif_rand() produces a uniform random number.

in Randall [11]). As the amount of computational time required for an ACS iteration is different to that of a PHIL search iteration, approximately the same amount of computational time per run is given to both strategies. This is based on 3000 ACS iterations. It must be noted that the ACS solver applies a standard local search (using inversion as the operator) to each solution that is generated.

Ten TSP problem instances are used to test both the ACS strategy and PHIL search. These problems are from TSPLIB [13] and are given in Table 1.

Table 1. Problem instances used in this study

Name	Size (cities)	Best-Known Cost
hk48	48	11461
eil51	51	426
st70	70	675
eil76	76	538
kroA100	100	21282
bier127	127	118282
d198	198	15780
ts225	225	126643
pr299	299	48919
lin318	318	42029

4.2 Results and Comparison

The results for the ACS and PHIL search strategies (in terms of objective cost and the amount of computational time required to reach a run's best objective value) are given in Tables 2 and 3 respectively. In order to describe the range

of costs gained by these experiments, the minimum (denoted "Min"), median (denoted "Med") and maximum (denoted "Max") are given. Non-parametric descriptive statistics are used as the data are highly non-normally distributed. Additionally, each cost result is given by a relative percentage difference (RPD) between the obtained cost and the best known solution. This is calculated as $\frac{E-F}{F} \times 100$ where E is the result cost and F is the best known cost.

Table 2. The results of the ACS strategy on the TSP instances. Note that Runtime is recorded in terms of CPU seconds

Problem	Cost (RPD)			Runtime		
	Min	Med	Max	Min	Med	Max
hk48	0	0.08	0.08	0.04	1.29	16.32
eil51	0.47	2	2.82	0.08	0.49	40.69
st70	0.15	1.33	2.07	36.39	43.48	87.56
eil76	0.19	1.3	2.42	0.08	70.23	114.73
kroA100	0	0	0.54	8.67	34.58	192.17
bier127	0.32	0.72	1.87	58.64	253.21	855.28
d198	0.16	0.33	0.6	154.53	1723.34	2422.52
ts225	0.63	1.15	1.93	513.65	3019.9	5484.59
pr299	0.42	0.92	2.68	10139.87	10794.69	13470.37
lin318	1.39	1.92	3	10388.72	14185.36	16090.43

Table 3. The results of the PHIL search strategy on the TSP instances

Problem	Cost (RPD)			Runtime		
	Min	Med	Max	Min	Med	Max
hk48	0	0.25	0.44	3.89	31.38	53.01
eil51	0	0.7	1.64	1.74	22.91	48.37
st70	0.15	0.3	0.74	19.73	127.04	264.78
eil76	1.12	2.42	3.35	56.7	138.69	309.24
kroA100	0.05	0.44	0.84	7.92	466.59	714.43
bier127	0.66	1.57	1.76	12.92	204.48	304.76
d198	1.12	1.66	1.86	17.26	1213.02	2172
ts225	0.34	0.61	0.93	173.25	2570.73	3602.72
pr299	2.13	2.64	3.7	455.17	6479.34	13885.99
lin318	2.96	3.86	4.51	5423.68	14961.38	19807.22

Given that PHIL search is a new technique, its overall performance is good in terms of solution quality and consistency. Both strategies can find solutions in all cases within a few percent of the best known costs. For the larger problems, PHIL search's performance is slightly behind that of ACS. However, it must be bourne in mind that this ACS (as is standard with ant colony techniques) also

executes local searches for each solution that it constructs. It is suspected that a greater exploration of the mechanics and the parameters of the new technique will yield still better results. This is discussed in the next section.

5 Conclusions

A new meta-strategy search technique, based on local search, has been proposed in this paper. PHIL search uses a recursive branching strategy, based on previous points within a search trajectory, to generate new searches. The advantage to this technique is that the branching strategy is computationally light in comparison to the characteristic mechanics of other meta-heuristics, particularly TS and ACO. Additionally, it only requires one parameter. The performance of PHIL search on benchmark TSP instances is encouraging. It can achieve solution costs within a few percent of best known costs and it is comparable to an ACS implementation.

In principle, PHIL search can be applied to any combinatorial optimisation problem that has been solved by traditional techniques (such as SA, TS and ACO). The development of the procedure is still in the initial stages. Some of the larger issues include the mechanics of the branching strategy and PHIL search's performance on a wider range of COPs. The former will involve the investigation of alternative strategies such as those based on heuristic strategies rather than just probabilities. As for the latter, the performance of PHIL search needs to be benchmarked against other meta-heuristics, especially on larger and more difficult problems. Of interest will be the incorporation of constraint processing within the strategy. Additionally, it is also possible to replace the local search branches with either tabu searches or simulated annealing.

References

1. Battiti, R., Tecchiolli, G.: The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. Technical Report UTM 432, Department of Mathematics, University of Trento (1994)
2. Battiti, R., Tecchiolli, G.: Learning with first, second and no dervatives: a case study in high energy physics. Neurocomputing **6** (1994) 181–206
3. Crauwels, H., Potts, C., van Wassenhove, L.: Local search heuristics for the single machine total weighted tardiness scheduling problem. INFORMS Journal on Computing **10** (1998) 341–350
4. Dorigo, M.: Optimization, Learning and Natural Algorithms. PhD. thesis, Politecnico di Milano (1992)
5. Ernst, A., Krishnamoorthy, M.: Solution algorithms for the capacitated single allocation hub location problem. Annals of Operations Research **86** (1999) 141–159
6. Feo, T., Resende, M.: Greedy randomised adaptive search procedures. Journal of Global Optimization **51** (1995) 109–133
7. Ghosh, D., Sierksma, G.: Complete local search with memory. Journal of Heuristics **8** (2002) 571–584

8. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston, MA (1997)
9. Mills, P., Tsang, E., Ford, J.: Applying an extended guided local search to the quadratic assignment problem. Annals of Operations Research **118** (2003) 121–135
10. Paquete, L., Stützle, T.: An experimental investigation of iterated local search for colouring graphs. In Cagnoni, S., Gottlieb, J., Hart, E., Raidl, G., eds.: Proceedings of EvoWorkshops 2002. Volume 2279 of Lecture Notes in Computer Science., Springer Verlag (2002) 122–131
11. Randall, M.: A systematic strategy to incorporate intensification and diversification into ant colony optimisation. In: Proceedings of the Australian Conference on Artificial Life, Canberra, Australia (2003)
12. Randall, M., Abramson, D.: A general meta-heuristic solver for combinatorial optimisation problems. Journal of Computational Optimization and Applications **20** (2001) 185–210
13. Reinelt, G.: TSPLIB - A traveling salesman problem library. ORSA Journal on Computing **3** (1991) 376–384
14. van Laarhoven, P., Aarts, E.: Simulated Annealing: Theory and Applications. D. Reidel Publishing Company, Dordecht (1987)
15. Voudouris, C.: Guided Local Search for Combinatorial Optimisation Problems. PhD. thesis, Department of Computer Science, University of Essex (1997)
16. Yuret, D., de la Maza, M.: Dynamic hill climbing: Overcoming the limitations of optimization techniques. In: The 2nd Turkish Symposium on Artificial Intelligence and Neural Networks. (1993) 208–212