TUNING OF HEURISTICS FOR TSP

M. Mojzeš, J. Kukal

Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague
Department of Software Engineering in Economics
Trojanova 13, 120 00 Prague, Czech Republic

Abstract

The heuristic algorithm performance evaluation may be a tricky task. The paper introduces an approach on how to address the problem of evaluating heuristic algorithm performance and thus enabling true tuning of these algorithms with respect to a given task.

1 Introduction

The solving of an optimization problem often requires application of a heuristic algorithm. There is a plenty of such algorithms and typically there are even more options of how to set up the parameters of a specific heuristic. Coherently with the theorem of "No Free Lunch" [1], each combination of heuristic algorithm and its settings can perform distinctively on different optimization problems. Besides, there is not just one criterion for the evaluation of heuristic performance itself.

Hence, when when developing a new algorithm or utilising an existing one to solve an *optimization problem* (OP), it is relevant to question *how to compare different heuristics and/or their parameter settings* when searching for the solution a specific OP.

A few attempts have been made [2] [3] to point out the difficulties one will deal with when analyzing and comparing heuristic algorithms performance, or to describe some of the "best practices" in this field, but a formalized framework is missing and we do believe that a contribution to this field could be made.

In our previous work [7] we have introduced a set of techniques we find useful for the analysis of the heuristic algorithms performance. These techniques can be divided into two main categories - *deterministic* and *stochastic*. Apart from the principles on which these methods are based, they are both delivering slightly different results. Using the deterministic methods, we arrive at a precise ranking of the heuristics performance, i.e. we know which has the best performance, which is the second one, etc. On the other hand, stochastic methods can generate a "cluster" of the best performing heuristics. Every heuristic in this cluster is better than the rest and heuristics in the cluster are all equally good by the means of statistical analysis.

Nevertheless, in this paper, we are focusing only on the deterministic methods and we are proposing an approach that enables also the deterministic methods to produce results similar to the latter category. Last, but not least, we will also demonstrate the use of this approach.

2 Methodology of comparison

Before detailed description of the proposed techniques, we begin with a more exact problem definition and the definition of heuristics performance measures.

An OP can be defined as minimization of the objective function f: $\mathbf{D} \to \mathbb{R}$ where

$$\mathbf{D} = \{\mathsf{x} \in \mathbf{X}^n \mid \mathsf{a} \le \mathsf{x} \le \mathsf{b}\}$$

is an appropriate domain. For purposes of this paper we are using the binary domain, $\mathbf{X}^n = \{0,1\}^n$, but it can be an integer or a real one as well. Let's suppose that we have an acceptable value of the objective function f^*. Then we can define a set of solutions, the goal set, as

$$\mathbf{G} = \{\mathbf{x} \in \mathbf{D} \mid \mathrm{f}(\mathbf{x}) \leq f^*\}$$

where

$$f^* \geq \min\{\, \mathrm{f}(\mathbf{x}) \mid \mathbf{x} \in \mathbf{D}\,\}.$$

We will suppose that we have a set of $H \in \mathbb{N}$ heuristics, each having $P_i \in \mathbb{N}$ parameters for $i = 1, 2, \ldots, H$. Then $p_{i,j} \in \mathbf{P}_{i,j}$ means that the setting of j-th parameter of i-th heuristic $p_{i,j}$ belongs to domain $\mathbf{P}_{i,j}$ which is a set of any distinct values specific for the given heuristic algorithm implementation, e.g. real numbers, logical values, or text strings. Domain $\mathbf{P}_{i,j}$ has the cardinality (number of elements) $C_{i,j} = \mathrm{card}(\mathbf{P}_{i,j})$. Actual parameter settings and their combinations may be based on recommendations (e.g. [6]), own experience, or by "random shooting" in the worst case.

2.1 Performance measures

Following the principles of Monte Carlo simulation we will run every *instance* of heuristic algorithm (meaning one specific heuristic approach with one specific setting of its parameters) for a specified, sufficiently large, number of times $q \in \mathbb{N}$ and then we will define the following estimates:

- *Reliability*, $REL = m/q$ where $m \in \mathbb{N}$ is the number of successful runs i.e. runs during which the algorithm found a solution from the goal set before exceeding the maximum allowed number of objective function evaluations (clearly $m \leq q$ and thus $REL \in [0, 1]$);
- *Mean Number of Evaluations*, $MNE = \frac{1}{m}\sum_{i=1}^{m} NE_i$ where $NE_i \in \mathbb{N}$ is the number of evaluations of the objective function until the algorithm found a solution from the goal set;
- *Standard deviation of the Number of Evaluations*, $SNE = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(NE_i - MNE)^2}$.

It may happen, that reliability will be so low that it is impossible to evaluate SNE and for that reason it is necessary to discard such instances from further analysis. Also, since reliability is often a very sensitive attribute, we recommend further preliminary elimination of very unreliable instances, no matter how effective they are.

The number of evaluations is a positive integer stochastic variable which significantly varies in order. It is hardly possible to suppose that this variable has a distribution which is close to the Gaussian normal one. It is a good habit to analyze its natural or decadic logarithm instead of the original value to eliminate positive skewness [2]. So we introduce the logarithmic measures as follows:

- *Logarithm of Number of Evaluations*, $LNE_i = \ln NE_i$ for $i \in 1, 2, \ldots, m$;
- *Mean Logarithm of the Number of evaluations*, $MLN = \frac{1}{m}\sum_{i=1}^{m} LNE_i$;
- *Standard deviation of the Logarithm of Number of evaluations*, $SLN = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(LNE_i - MLN)^2}$.

Our task is to maximize REL and minimize MLN and SLN together. However, at this point it is obvious that REL, MLN and SLN are independent variables, and, typically, they are in contrast with each other (e.g. a very reliable heuristic has higher MLN than another, less reliable one). Therefore this is a typical example of a *multi-criteria decision analysis.*

We can start analysing weakly performing instances using the condition of *Pareto optimality* [4] – this way we can exclude heuristics which are worse than others in every measured characteristic. Nevertheless, in most cases this will not be of great impact and there will still be plenty of instances to choose from.

2.2 Proposed methodology

The *weighted approach*, which is discussed in this paper, is based on minimizing the general formula with positive weights $w_i \in \mathbb{R}$ for $i \in 1, 2, 3$ of included performance measures:

$$\mathrm{F} = w_1 \cdot MLN + w_2 \cdot SLN + w_3 \cdot LNR$$

where $LNR = -\ln REL$.

Weights can be determined using various techniques. Some of them are motivated by the traditional Feoktistov's criterion: $FEO = MNE/REL$ [5].

However, we are suggesting a criterion that works on a presumption that NE can be approximated by the *exponential distribution* with time constant T [7] and it is in the form of

$$F = MLN + \frac{C \cdot \sqrt{6}}{\pi} \cdot SLN + LNR$$

where C is the Euler's constant and just for illustration $\frac{C \cdot \sqrt{6}}{\pi} \cong 0.4501$.

Moreover, to be able to decide which instance of heuristic algorithm is *efficient* and which one is not, we are suggesting certain bounds of acceptance. We say that the instance k is efficient when the value of its criterion F_k satisfies the following inequality:

$$F_k \leq F_{\min} + \frac{\pi}{\sqrt{3q}} \cdot \Phi^{-1}(1 - \alpha)$$

Here, $F_{\min}$ is the score of the best instance and Φ^{-1} is inverse of cumulative distribution function (cdf) of the normal distribution $\mathrm{N}(0, 1)$.

Previous formula is based on pessimistic assumption that random variable F has a variance $\sigma_F^2 = \frac{\pi^2}{6q}$ which is q times smaller than variance in the case of random shooting. Now we can test the hypothesis $\mathrm{H}_0 : F_k = F_{\min}$ against $\mathrm{H}_1 : F_k \neq F_{\min}$ at the level of significance α.

Under this assumptions the variable $F_k - F_{\min}$ follows normal distribution $\mathrm{N}(0, 2\sigma_F^2)$ and after substitution:

$$F_k - F_{\min} \sim \mathrm{N}(0, \frac{\pi^2}{3q})$$

In the case of symmetric testing, the hypothesis is rejected when

$$|F_k - F_{\min}| > \frac{\pi}{\sqrt{3q}} \cdot \Phi^{-1}(1 - \frac{\alpha}{2})$$

However, we need to test whether $F_k < F_{\min}$ and thus reject the hypothesis if

$$F_k > F_{\min} + \frac{\pi}{\sqrt{3q}} \cdot \Phi^{-1}(1 - \alpha)$$

and, for $\alpha = 0.05$, it is approximately

$$F_k > F_{\min} + \frac{2.9834}{\sqrt{q}}$$

3 Case study

3.1 Testing environment

The *Travelling Salesman Problem* (TSP) we have implemented consists of searching for the shortest route between cities on an integer grid, where paths between all possible pairs of cities exist and respective distances are calculated using the Euclidean distance. One can quickly figure out that if we mark $a \in \mathbb{N}$ and $b \in \mathbb{N}$ the width and height of our grid, then the shortest length equals $a \cdot b + \sqrt{2} - 1$ if both a and b are odd numbers or $a \cdot b$ otherwise. This way we can set the f^* value, since the total length of our path can be the objective function. However, for a computer this is not an easy task to solve.

While on the other hand we have the heuristic algorithms to deal with the testing problem. The first algorithm we have used is the *Genetic Optimization* (GO) [6], and since its detailed description is out of scope of this paper, we will mention only the parameters and respective settings which are to be analysed:

- $N \in \{5, 50, 100\}$ – the size of population,
- $T_{\mathrm{sel}} \in \{0.1, 10, 100\}$ – temperature controlling the probability of selection,
- $R \in \{0.001, 0.1\}$ – radius of mutation,
- $rep \in \{1, 3, 5\}$ – number of generation repetitions,
- $gdc \in \{0, 1\}$ – indication as to whether gradually decrease T_{sel} and R over time.

Finally, our implementation of the *Fast Simulated Annealing* (FSA) [6], the second heuristic, has a following set of parameters and settings:

- $T_0 \in \{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10\}$ – initial temperature,
- $n_0 \in \{1, 10, 500, 1000\}$ – cooling delay,
- $\alpha \in \{0.5, 1, 1.5, 2\}$ – cooling exponent.

To summarize the number of instances, we get $NI = 3 \cdot 3 \cdot 2 \cdot 3 \cdot 2 + 7 \cdot 4 \cdot 4 = 108 + 112 = 220$. The first part of the sum is the product of individual distinct parameter values counts of GO and the second one of FSA, which performance we want to compare.

3.2 Results

To get a satisfactorily precise result, we set $q = 100$, and thus we analyzed $NI \cdot q = 22\,000$ of single runs of heuristic algorithms. Table 1 gives us a rough idea of what the outcome of our experiment was.

Table 1: Overall performance results

	GO	**FSA**	**Total**
All	108	112	220
Reliable	83	112	195
Effective	0	40	40
The best one	0	1	1

Note: A *reliable* instance is the one having $REL \geq 0.2$.

Even from this, very high-level, statistic we can draw interesting conclusions about the two different heuristics: FSA, as opposed to GO, is convincingly the *best* heuristic for this problem. Not only all of its instances were reliable and more than 35 percent of them were effective, but in fact all of the effective instances were generated solely by FSA.

Table 2 supports the above mentioned conclusions and adds also another relevant observations, which are rather self-explanatory.

Table 2: Basic performance characteristics

Characteristic	**GO**	**FSA**
Average reliability	0.72	0.82
Average MNE	1559.1	450.1
Average SNE	1380.4	585.9
$F_{\min}$	6.23	5.43
F_{avg}	7.29	6.02
$F_{\max}$	10.16	7.61

It is also worth mentioning that eight instances of GO had $REL \leq 0.01$. This prevented us from calculating the F criterion value altogether.

Last, but not least, we are presenting a detailed overview of the effective instances parameter settings in table 3.

Table 3: Effective instances parameter settings

Heuristic	**Parameter**	**Value**	**Times present**	**Impact ratio**
FSA	T_0	0.0001	12	0.300
FSA	T_0	0.00001	11	0.275
FSA	T_0	0.001	6	0.150
FSA	T_0	0.01	6	0.150
FSA	T_0	0.1	2	0.050
FSA	T_0	1	2	0.050
FSA	T_0	10	1	0.025
FSA	n_0	1	13	0.325
FSA	n_0	10	12	0.300
FSA	n_0	500	9	0.225
FSA	n_0	1000	6	0.150
FSA	α	1.5	12	0.300
FSA	α	2	10	0.250
FSA	α	0.5	9	0.225
FSA	α	1	9	0.225

A table similar to table 3 may be of significant use, since it does reveal the parameters and their settings which have the biggest impact on the performance of the heuristic. E.g. from the values in this specific one we could conclude that it may be a good idea to pick T_0 from $\{0.0001, 0.00001\}$, n_0 from $\{1, 10\}$ and α from $\{1.5, 2\}$ as these settings have notably higher impact ratio than the remaining ones.

For the sake of illustration, we are also including an analogous table for the GO heuristic (table 4), but of course, under the assumption that it would be the only available heuristic to deal with the given problem.

Table 4: Effective instances parameter settings - GO only

Heuristic	**Parameter**	**Value**	**Times present**	**Impact ratio**
GO	N	5	6	0.400
GO	N	100	5	0.333
GO	N	50	4	0.267
GO	T_{sel}	10	11	0.733
GO	T_{sel}	0.1	2	0.133
GO	T_{sel}	100	2	0.133
GO	R	0.1	11	0.733
GO	R	0.001	4	0.267
GO	rep	1	7	0.467
GO	rep	3	6	0.400
GO	rep	5	2	0.133
GO	gdc	0	11	0.733
GO	gdc	1	4	0.267

In such case we may proclaim that good settings of GO are $T_{sel} = 10$, $R = 0.1$, $rep \in \{1, 3\}$ and $gdc = 0$. However, further search for the optimal setting of N might be irrelevant as it will most probably not bring the desired effect.

However, here we are analysing the effects of possible tuning of individual settings. Undoubtedly, the settings, and most importantly the effects they may have on the performance, are not completely independent – so it is here, where we see the most straight-forward potential for future work, to analyse effects of settings combinations.

4 Conclusions

Using the proposed approach on a specific instance of Travelling Salesman Problem we may conclude that our implementation of heuristic algorithm of Fast Simulated Annealing significantly outperformed the one of Genetic Optimization.

In general, the above-mentioned conclusion is an outcome of the technique that is developed from a traditional way of measuring the heuristic algorithm performance and deterministic multi-criteria decision analysis methods. We do believe that the conclusion is valid also as a feasibility demonstration of such approach and this can be used to better understand and interpret the behaviour of a heuristic and reveal the potential effect of settings tuning.

Acknowledgement: This paper was created under the support of grant OHK4-165/11 CTU in Prague.

References

[1] Wolpert D. H., Macready W. G., No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, 1997

[2] McGeoch C. C., Experimental Analysis of Algorithms, *Handbook of Global Optimization Volume 2*, Kluwer Academic Publishers, 2002, pp. 489–513

[3] Battiti R., Machine Learning Methods for Parameter Tuning in Heuristics, *5th DIMACS Challenge Workshop: Experimental Methodology Day*, Rutgers University, 1996

[4] Van Veldhuizen D. A., Lamont G. B., Evolutionary Computation and Convergence to a Pareto Front, *Proceedings of the 3rd Annual Conference on Genetic Programming*, Stanford University, San Francisco CA, 1998, pp. 221–228

[5] Feoktistov V., *Differential Evolution: In Search of Solutions*, Springer, 2006

[6] Kvasnička V., Pospíchal J., Tiňo P., *Evolutionary Algorithms* (in Slovak), STU Bratislava, 2000

[7] Mojzeš M., Kukal J., Tran V. Q., Jablonský J., Performance Comparison of Heuristic Algorithms via Multi-Criteria Decision Analysis *Proceedings of Mendel 2011 Conference*, Brno University of Technology, Brno, 2011, pp. 244–251

Matej Mojzeš
mojzemat@fjfi.cvut.cz

Jaromír Kukal
jaromir.kukal@fjfi.cvut.cz