Introducing a Hash Function for the Travelling Salesman Problem for Differentiating Solutions

Mehdi El Krari
Université de Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL
Lille, France
mehdi@elkrari.com

Rym Nesrine Guibadj
Université du Littoral Côte d'Opale, EA 4491 - LISIC
Calais, France
rym.guibadj@univ-littoral.fr

John Woodward
Operational Research Group, School of Electronic Engineering and Computer Science, Queen Mary University of London
London E1 4NS, UK
j.woodward@qmul.ac.uk

Denis Robilliard
Université du Littoral Côte d'Opale, EA 4491 - LISIC
Calais, France
denis.robilliard@univ-littoral.fr

ABSTRACT

Solution comparison is a process used in different classes of evolutionary algorithms. It can be either for choosing a better solution or differentiating a pair of solutions. While there is no doubt that the fitness function allows determining the best solution, its use to distinguish between solutions is questionable, especially for combinatorial optimisation problems. This short paper focuses on the Travelling Salesman Problem (TSP). 39 instances of the TSPLIB are chosen, two solution samples are generated for each instance. A collision analysis of the fitness function one the TSP is presented, then an introduction to an efficient hash function with almost zero collisions.

CCS CONCEPTS

• **Computing methodologies** → **Discrete space search**; Randomized search; • **Applied computing** → *Operations research*;

KEYWORDS

Hash functions, Combinatorial Problems, Travelling Salesman Problem

ACM Reference Format:
Mehdi El Krari, Rym Nesrine Guibadj, John Woodward, and Denis Robilliard. 2021. Introducing a Hash Function for the Travelling Salesman Problem for Differentiating Solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference 2021 (GECCO '21)*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459580

1 INTRODUCTION

For each Combinatorial Optimisation Problem (COP), an evaluation function joins a fitness measure to each solution. Such function should be defined to differentiate a pair of solutions based on their respective quality, but also to direct the search process. These two points are expected to be reached by a single function [1].

Mapping different solutions of a search space to the same fitness value is observed on different COPs. Distinguishing between solutions only with their fitness values is therefore impossible for a metaheuristic, and this lack of information may obstruct some of its characteristics.

Trajectory-based metaheuristics [6] may suffer from the repetitions of the fitness values over an instance, especially when a value is redundant in a region of the search space. A *plateau* is then formed and can be a trap for a metaheuristic since the fitness value, which is the same, does not provide any additional information for the search process.

Population-based algorithms [2] are also affected by the redundancy of fitness values. Since keeping the population with a maximum diversity is pointed out in each Genetic Algorithm (GA), it is measured based on the fitness value of each individual. Indeed, a subset of different solutions with the same fitness value can mislead the algorithm to a biased converge rate and lead to premature convergence.

We introduce in this short paper a new hash function for the Travelling Salesman Problem (TSP) as a fair alternative to differentiate between a pair of solution. An empirical study on the fitness function is provided prior to our hash function to show the high number of repetitions for each fitness value on the same instance, which can't be with no effect on the evolutionary process of an algorithm.

2 EMPIRICAL STUDY OF FITNESS FUNCTION'S COLLISIONS OF THE TSP

Using the fitness function (f_{fit}) to differentiate solutions within a search space is a common practice in evolutionary computation. The repetition of a fitness value over a set of solutions is usually observed on different COPs. But no prior work points out how much the fitness values are redundant on a given instance.

39 instances are chosen from the TSP benchmark TSPLIB [3] (sizes n range from 51 to 575). Two samples of distinct solutions are generated for each one. The first sample (S_{LO}) is a set of n^2 local

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459580

optima obtained with an ILS framework [4]. The second one (S_{rand}) contains $10 \times n^2$ random solutions.

As a preliminary step to measure the impact of using the fitness function as a comparison tool, the number of collisions is computed over the above-mentioned samples and shown in table 1. We count a collision between a pair of solutions s_1, s_2 when $f(s_1) = f(s_2)$. We expose in this short paper 10 instances from the 39 studied. We list for each one, and for each sample (S_{LO} and S_{rand}), the sample size $|S|$, and the number of collisions C, and the number of the different fitness values Fit found in each sample. The last column of each part of the table is discussed in the next Section.

Although the existence of collisions is, as we mentioned earlier, not unpredictable, the figures exposed in table 1 outpace our predictions. Indeed, a very high number of collisions is noticed in both local optima and random samples, with up to millions of collisions for the smallest ones. Local optima, who share common edges between them, can explain the first part of the table. We also notice, from Fit_{LO} and Fit_{rand}, that the generated samples are mapping to tiny sets of fitness values, making them very redundant. Moreover, according to Fit_{LO} and Fit_{rand}, we notice very small sets of fitness values to whom the solutions of S_{LO} and S_{rand} are mapping. In other words, the large set of solutions are distributed over a small set of fitness values, making some fitness values very repetitive.

Table 1: Fitness function collisions observed on large samples of TSP instances

Instance	$\|S_{\text{LO}}\|$	C_{LO}	Fit_{LO}	η	$\|S_{\text{rand}}\|$	C_{rand}	Fit_{rand}	η
eil51	2,634	126,875	53	0	26,010	1,068,752	577	2
st70	4,904	229,647	99	0	49,000	1,876,506	1,170	0
rat99	9,847	457,303	189	0	98,010	3,317,137	2,613	2
kroA100	10,007	30,902	2,163	0	100,000	170,251	32,619	1
ts225	50,649	128,149	12,854	0	506,250	824,742	170,884	0
lin318	101,375	2,219,290	3,951	0	1,011,240	10,188,376	78,223	0
rd400	160,396	18,439,021	1,398	0	1,600,000	81,553,529	29,261	0
fl417	174,140	28,258,436	1,424	1	1,738,890	31,432,278	80,618	0
pcb442	195,604	7,548,302	4,649	0	1,857,610	7,752,811	303,216	0
rat575	330,697	234,671,318	592	0	3,306,250	676,190,119	16,898	0

3 A HASH FUNCTION FOR FAIRER DIFFERENTIATION

The obtained results in the previous section were the main motivation to propose a new function to differentiate solutions as an alternative to the fitness function. Some hash functions have been proposed [5, 7] for COPs (specifically permutation-based ones) for specific uses. We briefly present in this section our hash function (η) with its main characteristics.

Like any other mathematical function, operands and operators had to be set. We decide to design η with no random values or vectors. Only instance data, which are the distance matrix and the set of nodes identifiers ($[1; n]$), will be the operands of the function. Three operators are composing η: addition, multiplication and modulo. The latter is slightly redefined in formula 1 as we are dealing with symmetric TSP. The *mod* operator will allow getting the same hash value whatever the reading direction and prevent zero values.

$$mod(a, b) = max(a, b)\%min(a, b) \tag{1}$$

We designed the function η as shown in formula 2. π_i is the identifier of the i^{th} node in the permutation π. $C = (c_{ij})$ is the distance matrix of the studied instance.

$$\begin{aligned} \eta = {} & mod(\pi_1, \pi_n) \times (\pi_1 + \pi_n) \times (\pi_1 \times \pi_n) \times c_{\pi_1, \pi_n} + \\ & \sum_{i=1}^{n-1} mod(\pi_i, \pi_{i+1}) \times (\pi_i + \pi_{i+1}) \times (\pi_i \times \pi_{i+1}) \times c_{\pi_i, \pi_{i+1}} \end{aligned} \tag{2}$$

We list in the last columns of the table 1 the number of collisions we obtained with our hash function η. We observe a consequential reduction of collisions compared to the fitness function. η succeeded to get zero collisions on 36 (resp. 32) instances for S_{LO} (resp. S_{rand}). The overall average for our hash function is less than one collision in each set of samples.

A hash function can then provide a fairer tool for different classes of evolutionary algorithms to distinguish solutions. Trajectory-based algorithms can detect whether a solution was really visited or not. While GAs will be allowed to determine the real convergence rate since each individual has its own hash value. Moreover, η can be embedded in a metaheuristic, with a constant time cost, without penalising the global complexity of the algorithm.

4 CONCLUSION

The important number of collisions observed on the fitness function and the significant repetitions in its values make it inappropriate to differentiate solutions. We introduced in this short paper a new effective hash function, with a very low number of collisions, reaching zero in almost all the samples of the studied instances.

To validate the harmful effect the fitness function may have on evolutionary algorithms, our next work will revisit different class of metaheuristics with and without hash functions. A positive effect is expected on both local search-based algorithms and GAs.

We focused till now on the TSP as one of the most studied COP in the literature. Other problems, with different solution representations, will be examined in the next works by adapting the proposed hash function η or by proposing new ones.

REFERENCES

[1] Alexander EI Brownlee, John R Woodward, and Jerry Swan. 2015. Metaheuristic design pattern: surrogate fitness functions. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation.* 1261–1264.

[2] P. Larrañaga, Cindy Kuijpers, R.H. Murga, I. Inza, and S. Dizdarevic. 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial intelligence review: An international survey and tutorial journal* 13, 2 (4 1999), 129–170.

[3] Gerhard Reinelt. 1991. TSPLIB—A traveling salesman problem library. *ORSA journal on computing* 3, 4 (1991), 376–384.

[4] Thomas Stutzle. 2006. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research* 174, 3 (2006), 1519 – 1539.

[5] Toffolo Túlio A.M., Vidal Thibaut, and Wauters Tony. 2019. Heuristics for vehicle routing problems: Sequence or set optimization? *Computers Operations Research* 105 (2019), 118 – 131.

[6] Christos Voudouris and Edward Tsang. 1999. Guided local search and its application to the traveling salesman problem. *European journal of operational research* 113, 2 (1999), 469–499.

[7] David L. Woodruff and Eitan Zemel. 1993. Hashing vectors for tabu search. *Annals of Operations Research* 41 (1993), 123–137.