

# Refining Optimization Problem Structure Notion: Traveling Salesman Problem is Strongly Structured

Benjamin Weinberg

Cyril Fonlupt

Denis Robilliard

Laboratoire d'Informatique du Littoral  
Maison de la Recherche Blaise Pascal  
50, rue Ferdinand Buisson - BP 719  
62228 CALAIS Cedex, FRANCE

weinberg@lil.univ-littoral.fr

fonlupt@lil.univ-littoral.fr

robillia@lil.univ-littoral.fr

**Abstract-** The goal of this article is to refine the notion of structure as proposed in [6]. More precisely, in that article, the authors discuss the scope of NFL theorem [7] on classes of problems and especially those known in combinatorial optimization as traveling salesman problem (TSP), Graph Coloring Problem (GCP) *etc.* For that, a notion of structure is defined using a set of configurations (namely core set). The fitness values of configurations of this set is enough to characterize the whole fitness function. In this paper, we clarify the structure notion in order to exhibit classes of problems on which NFL does not hold true. Then, we see that structures often occurs in combinatorial optimization. Although, this technique could be used for any class of problems with linear expression of fitness function, it may in many cases be impracticable due to its complexity. So, we introduce here a notion of strong structure building a core set in *polynomial time*. This was implicitly proved in [6] for Graph Coloring Problem. Moreover, we prove that Traveling Salesman Problem is strongly structured. In this case of study and contrary to the GCP, we can build a core set even if it does not allow explicit reconstruction of problem data.

**keywords:** No Free Lunch theorem, Theoretical approach, Structure, Traveling Salesman Problem

## 1 Introduction

Impossibility statements theorems appear in several fields of research. For instance, in computational logic and proof theory, there exists Gödel's theorem which states the existence of some unprovable true results. In Information Theory, the Shannon's theorem states there does not exist an universal compression algorithm. Despite of this, automatic theorem proving and data compression are still important research fields. In the optimization domain, there exists such a theorem called the No Free Lunch theorem (NFL) [7]. Roughly speaking, this theorem proves the lack of absolute dominance of any heuristic on the whole set of optimization problems. One of the most remarkable aspect of NFL is that this result is valid for blind methods such as the random algorithm which provides new configurations without matter to previous visited ones.

However when a new metaheuristic method is proposed,

it is generally applied on a specific class of problems and tested on several benchmarks. "Good" results are generally provided and we hope that the algorithm can be reused for solving other problems or at least some lessons can be drawn from it. Given the NFL theorem, one can wonder why those algorithms should perform well on a given testbed optimization problem and wonder why new heuristics and metaheuristics are regularly designed.

Indeed, everyone can test that, in combinatorial optimization, a simple hill climbing algorithm does not provide so bad results when compared to random search. The relative efficiency of the first method compared to the second one seems to contradict NFL theorem.

Recently, Weinberg and Talbi proposed a notion of structure qualifying classes of problems on which NFL does not hold [6]. and we deepen this study on the case of Traveling Salesman Problem, proposing a refined notion called *strong structure*.

This article is composed as follows. In the first section, we recall some related works on NFL. In the second section, we see that many combinatorial optimization problems can be qualified as structured problems. We illustrated this on the case of the TSP. Then we introduce the notion of *strong structure* for a class of problem, in the third section. This notion is effective than the simple structure notion defined in [6]. In the fourth section, we prove that TSP complies with property of *strong structure*. In the fifth section, we discuss the scope of this results. We conclude and suggest some future works in the last section.

## 2 Related works

The development of software platforms that allow an easy instantiation of metaheuristics [5], suggests that there exists something sharable between metaheuristics. They are usually implemented using the object oriented paradigm and allow the development of dedicated components. In such a platform, metaheuristics are seen as frameworks and users have to specify at least the objective function, and possibly the coding of the configurations (i.e. solutions or points of the search space) and the search operators.

In [1], Culberson comments the NFL theorem and points out that we have to sort out different cases based on the "knowledge" used by the algorithm with regards to the

problem. Hereafter, we recall the five cases (also named scenarii) defined by Culberson. As Culberson did, they are illustrated using graph coloring problem (GCP):

1. **Full Knowledge:** the algorithm is developed for solving one specific problem. The full graph is known by the algorithm and it can use all its specificities to find the best coloring. It is tolerated that the algorithm can be inefficient on another graph.
2. **Partial Knowledge:** the algorithm knows how the objective function is described. The algorithm knows that there exists a (unknown) graph and knows its number of nodes. An efficient coloring algorithm must find quickly a good (not necessary the best) coloring for the overall set of graphs.
3. **Little Knowledge:** the algorithm knows that it optimizes some problems from a large class of problems like NP-hard ones.
4. **Almost no Knowledge:** the algorithm knows that the objective function is not so heavy to compute. Droste *et al.* define this case that the Kolmogorov complexity [2] of the fitness function is low. But in this scenario, the algorithm cannot suppose that it optimizes such or such (class of) problem(s).
5. **No Knowledge:** the algorithm minimizes an objective function.

In 2003, Woodward and Neil [8] begin to list three limitations for NFL use:

1. **revisiting points**, NFL hypothesis assume that the configuration may be visited only once. This is unpractical. Indeed algorithms cannot memorize all visited configurations, without requiring huge memory amounts.
2. **all functions**, NFL hypothesis assume all functions<sup>1</sup>. One can wonder what happens to the NFL result if there is a restriction on the set of function. This argument is developed in the remainder of this paper and we will study it thoroughly.
3. **overheads**. Time complexity is not focus, in NFL assumptions. As said in the first point, memorizing the visited configurations implies a huge overhead due to the size of search spaces. But some other phenomena may change the complexity of fitness computation of a new configuration, As adaptive behavior of the heuristic or the incremental computation of fitness.

Weinberg and Talbi [6], Igel and Toussaint [3] and Schumacher [4] deal with limitations of the use of NFL theorem. Their approaches share a common point; they discuss on the set of problems considered by NFL assumption.

Schumacher [4] proves that the result of NFL holds if and only if the set of function is closed under permutation

<sup>1</sup>in its first formulation; other studies talk about more restrictive but special sets of functions.

(c.u.p.). More precisely, a set  $\mathcal{F}$  of optimization problems is said to be c.u.p. if  $\mathcal{F}$  verify the following property:

$$\forall f \in \mathcal{F}, \forall \sigma : \mathcal{X} \rightarrow \mathcal{X}, \sigma \text{ bijective}, f \circ \sigma \in \mathcal{F}$$

where  $\mathcal{X}$  is the search space.

It may not be easy in general to affirm or infirm that this property is true on a specific set of problem.

Igel and Toussaint [3] moderate the scope of NFL theorem for a class of functions having special neighborhood operators. As there is often a correlation between the fitness of two neighbors, thus the authors show the importance of the choice of operators when a metaheuristic is designed.

Weinberg and Talbi propose an approach dedicated to metaheuristic [6]. They define a notion of structure for a class of problems. On such a set of problems, NFL result does not hold. This is what we will deepen here. We recall hereafter the definition of structure for classes of problems.

**Definition 1** We say that a class of problems is structured if there exists an algorithm building a strict subset of configurations, such that if the fitness of those configurations are known then the fitness of all configurations can be deduced.

To simplify our discussion, we also use the following definition.

**Definition 2** We call such a set of configurations a core set.

First, we notice these definitions cannot exactly match NFL requirements. Indeed, this definition was first written to squeeze the proof of NFL<sup>2</sup>. In order to be sure that NFL result does not hold on a structured set, we can, for example, verify that two configurations *that are not into* the core set have different fitness for at least one observed problem.

If both properties are verified, one can see that the studied set of problems are not c.u.p. Indeed, let  $f$  be a function of a structured class  $\mathcal{F}$ , let  $\mathcal{C}$  be a core set and let  $c_1$  and  $c_2$  be two configurations not in  $\mathcal{C}$  such that  $f(c_1) \neq f(c_2)$ . Let us notice  $\tau_{c_1 c_2}$  the mapping defined by :

$$\begin{aligned} \tau_{c_1 c_2} : \mathcal{X} &\rightarrow \mathcal{X} \\ c &\mapsto \tau(c) = \begin{cases} c_2 & \text{if } c = c_1 \\ c_1 & \text{if } c = c_2 \\ c & \text{otherwise} \end{cases} \end{aligned}$$

$\tau_{c_1 c_2}$  is bijective.  $f \circ \tau_{c_1 c_2}(c) = f(c)$  for all  $c \in \mathcal{C}$ . As  $f \circ \tau_{c_1 c_2}(c_1) = f(c_2) \neq f(c_1)$  so  $f \circ \tau_{c_1 c_2} \notin \mathcal{F}$ . So,  $\mathcal{F}$  is not c.u.p. .

### 3 About the structure of TSP

In this section, we examine the case of the TSP structure. We can have the same reasoning for many combinatorial optimization problems as long as we succeed in expressing the fitness function as a linear combination of variables.

A TSP is given by:

- $N$  : the number of cities;
- $D$  : the distance matrix (i.e. matrix  $N \times N$  provides the distance between each pair of cities). In classical TSP problem,  $D$  is supposed to be symmetrical.

<sup>2</sup>Weinberg and Talbi exhibit how the proof assumes that the studied set of fitness functions is not structured.

The goal is to provide a Hamiltonian cycle on the cities (also called a tour) with the shortest length. The search space is the set of all Hamiltonian cycles:

$$\{\sigma : \{2 \dots N\} \rightarrow \{2 \dots N\} \mid \sigma \text{ is bijective}\}$$

The length of a tour is given by the following function :

$$f(\sigma) = \sum_2^{N-1} D(\sigma(i), \sigma(i+1)) + D(1, \sigma(2)) + D(\sigma(N), 1)$$

On GCP, Weinberg and Talbi, found a core set such that they could retrieve the problem instance description from it. As we will see below, we find a core set for the TSP, Although we cannot retrieve the whole distance matrix  $D$  from it. Nonetheless this core set, together with the existence of two outside configurations with different fitness are enough to prove that TSP is structured and thus does not fill the requirements for NFL theorem (see section 6).

The first step is to turn a configuration  $\sigma$  into a vector of 0 and 1:  $C = (c_{i,j})_{i \in \{1 \dots N\}, j \in \{i+1 \dots N\}}$ . Where  $c_{i,j} = 1$  if the tour  $\sigma$  uses the edge between cities  $i$  and  $j$ ;  $c_{i,j} = 0$  otherwise. We can define the function  $\tilde{f}$  defined by  $\tilde{f}(C) = \sum_{i,j} c_{i,j} \times D(i,j)$  and  $\tilde{f}(C) = f(\sigma)$  where  $\sigma$  is the configuration generating  $C$ .

Then, we can build a linear system  $(S)$  with  $\frac{N \times (N-1)}{2}$  variables and  $\frac{(N-1)!}{2}$  lines. An example of such a system generated by a TSP with five cities is presented in Figure 1.

For  $N$  big enough ( $N > 3$ ), such a system has obviously more lines than columns. But we are sure the system is consistent due to the fact it comes from a real TSP.

**Property 1** *If the rank of this system is equal to the number of variables then we can build  $D$ .*

**Property 2** *If the rank of this system is less than the number of variables then the set of distance matrices consistent with  $S$  is a vector space of dimension  $K = \frac{N \times (N-1)}{2} - \text{rank}(S)$ . In other words, we can arbitrarily set values to  $K$  coefficients of a matrix  $D'$  and deduce the other coefficients of  $D'$  in order to be consistent with  $(S)$ . So, even if  $D'$  is different of  $D$  the fitness of tours are identical using  $D$  and  $D'$ .*

**Remark 1** *The rank of this system is the number of useful lines, i.e. the minimum number of elements of a core set.*

So, the question is: "How to find useful lines?" The answers can be found by using Gaussian elimination. This seems easy but if we look more precisely, we observe that  $(S)$  is huge. As, the algorithm to compute lines is in  $O\left(\left(\frac{(N-1)!}{2}\right)^3\right)$ , means an exponential complexity in time.

As a conclusion of this section, we sum up several main points. As explain above, we do not need to be able to extract the original data from a problem to prove a class of problems respects the notion of structure defined in [6]. Second, TSP respects structure notion. Finally, the way used to prove that result is general and we could be used for other problems. More precisely, one can prove by the same procedure that any problem using a linear function as fitness is structured. It may be necessary to reformulate the problem to clearly exhibit a linear definition; in this case, we

do not need to take into account constraints on variables, as we consider every valid configuration.

## 4 Toward a notion of strong structure

Despite the last point seen above, we do not forget the catastrophic complexity of this approach: There is a real difference with the case of GCP presented in [6]. Indeed, in that case the authors use an applicable technique to exhibit a core set. The procedure they follow is polynomial in time with respects to the problem instance size. In the previous section, we use an **non-effective** technique: for example, a problem of size 30 will face more than  $4 \times 10^{30}$  lines in the system! It becomes non-effective to build such a system; even less to solve it by Gaussian elimination. So, we propose a more useful notion of structure.

**Definition 3** *We say that a class of problems is strongly structured if there exists an algorithm polynomial in time building a strict subset of configurations, so called a core set, such that if the fitness of those configurations are known then the fitness of all configurations can be computed.*

In the frame of strong structure, we also talk about a *core set*. Indeed the technique used to compute the core set is different but it verifies the same properties.

In the definition of strong structure, the term *polynomial* implies that the size of the core set is polynomial with respect to the size of the instances of the problem.

It seems obvious that the property of strong structure implies property of structure. Actually, Weinberg and Talbi proved that GCP is strongly structured. We can wonder if TSP is also strongly structured. In other words, can we find a polynomial way to compute a core set? The next section positively answers to this question.

## 5 Building a core set For TSP

In this section, we prove that TSP is strongly structured. In a first time, we prove by recurrence that for any  $N$  greater than or equal to three we have the following properties  $P_N$ :

**Definition 4** *Property  $P_N$*

1. *For a TSP  $T$  of  $N$  cities, the set of matrices generating the same TSP as  $T$  is a vector space of dimension  $N - 1$ .*
2. *There are  $\frac{(N-2) \times (N-1)}{2}$  linearly independent lines (tours) in the linear system associated to  $T$ .*

*In other words, We exactly know what is the rank of the system generated by a TSP problem and we effectively build a core set. In the second time, we deduce that TSP is strongly structured as corollary.*

**Lemma 1** *For all  $N \geq 3$ ,  $P_N$  holds.*

### 5.1 Initialization

To initiate the recurrence, we verify that  $P_3$  holds true. In this case we have three cities; So, only one tour. This obviously verify the second point: one line with at least a non-zero coefficient is obviously linearly independent. Now the

$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{3,4}$	$c_{3,5}$	$c_{4,5}$	
1	0	0	1	1	0	0	1	0	1	$\tilde{f}(C_1)$
1	0	1	0	1	0	0	0	1	1	$\tilde{f}(C_2)$
1	0	0	1	0	1	0	1	1	0	$\tilde{f}(C_3)$
1	1	0	0	0	1	0	0	1	1	$\tilde{f}(C_4)$
1	1	0	0	0	0	1	1	0	1	$\tilde{f}(C_5)$
1	0	1	0	0	0	1	1	1	0	$\tilde{f}(C_6)$
0	1	0	1	1	1	0	0	0	1	$\tilde{f}(C_7)$
0	1	1	0	1	0	1	0	0	1	$\tilde{f}(C_8)$
0	1	0	1	0	1	1	1	0	0	$\tilde{f}(C_9)$
0	1	1	0	0	1	1	0	1	0	$\tilde{f}(C_{10})$
0	0	1	1	1	0	1	1	0	0	$\tilde{f}(C_{11})$
0	0	1	1	1	1	0	0	1	0	$\tilde{f}(C_{12})$

On this example,  $C_k$  denote a configuration of the search space.

Figure 1: Example of System generated by a five cities TSP

first point, we can also linearly express the distance between the first and the second cities with regards to the two other distance. As  $D(1,2) = f(C) - D(2,3) - D(1,3)$  where  $f(C)$  is the fitness of the unique configuration.

## 5.2 Recurrence

Assume that for a given  $N(\geq 3)$   $P_N$  holds true for any TSP of  $N$  cities. Let  $A$  be a TSP with  $N + 1$  cities. Let  $(a_1 \dots a_{n+1})$  denote the cities of problem  $A$ . From  $A$ , we build a TSP  $B$  of only  $N$  cities denoted as  $(b_1 \dots b_n)$  as follows:

$$D(b_i, b_j) = \begin{cases} D(a_i, a_j) & \text{if } i \neq 1 \text{ or } j \neq n \\ D(a_1, a_{n+1}) + D(a_n, a_{n+1}) & \text{if } i = 1 \text{ and } j = n \end{cases}$$

Figure 2 illustrates the building of  $B$  from problem  $A$ . As  $D$  is symmetrical and we do not matter about elements on the diagonal, we have entirely defined problem  $B$ .

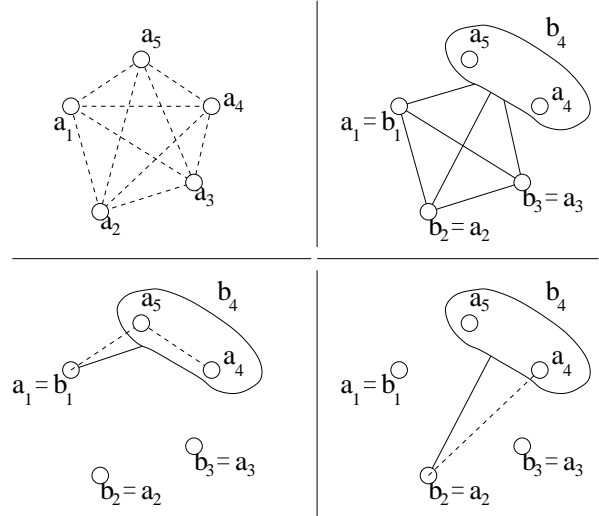
Now, as we know that  $P_N$  true on  $B$ . So, we obtain the core set for  $B$  which respects constraints of  $P_N$ . Let  $(v_1, v_2, \dots, v_{n-1})$  denote variables spanning the matrices equivalent to the matrix of  $B$ . Each  $D(b_i, b_j)$  can be compute as an affine function, denoted  $l_{b_i b_j}$  of  $(v_1, v_2, \dots, v_{n-1})$ :

$$D(b_i, b_j) = l_{b_i b_j}(v_1, v_2, \dots, v_{n-1})$$

Now we must find a affine parameterization of any  $D(a_i, a_j)$  using  $n$  variables.

First step, we choose the  $n^{th}$  variables. We must distinguish two cases:

- $D(b_1, b_n)$  is not one of the  $v_i$ . In other words, there exists an affine function  $l_{b_1 b_n}$  such that  $D(b_1, b_n) = l_{b_1 b_n}(v_1, \dots, v_{n-1})$ . We introduce a new variable  $v_n = D(a_1, a_{n+1})$ . We have  $D(a_n, a_{n+1}) = D(b_1, b_n) - D(a_1, a_{n+1}) = l_{b_1 b_n}(v_1, \dots, v_{n-1}) - v_n$ . We can affinely express  $D(a_n, a_{n+1})$  with  $(v_i)_{i \in \{1 \dots N\}}$ ;



In this example, full lines correspond to edges in problem  $B$  and dashed lines are edges from problem  $A$ . On both down schemas the length of the drawn edge of  $B$  is equal to the sum of the length of the drawn edge(s) of  $A$ .

Figure 2: Deduce smaller problem  $B$  from  $A$ .

- $D(b_1, b_n)$  is one of the  $(v_i)_i$ , namely  $v_k$ . In this case we have the choice to eliminate  $D(b_1, b_n)$  from all relation with a Gaussian elimination. Or we substitute  $v_k$  by  $v'_k + v_n$  in all relations; where  $v'_k = D(a_n, a_{n+1})$  and  $v_n = D(a_1, a_{n+1})$

As the variable names are dumb, we rename them  $(v_i)_{i \in \{1 \dots n\}}$  independently of the case.

**Remark 2** A tour in  $B$  having the edge  $(b_1, b_n)$  can be seen as a tour in  $A$  following a subpath  $(a_1, a_{n+1}, a_n)$  and reciprocally. So, we can deduce the fitness of those tours (in  $A$ ) with regard to  $(v_1 \dots v_{n-1})$  (without even using  $v_n = D(a_1, a_{n+1})$  (see figure 3). The fitness of those

configurations can be estimated directly from the problem  $B^3$ .

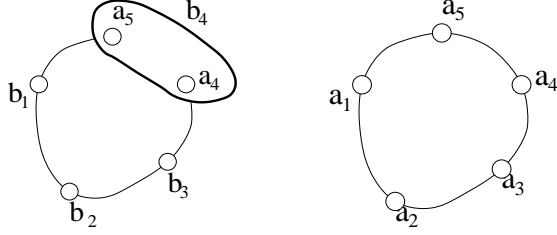


Figure 3: First tour which fitness is deductible.

Second step, we reuse every information directly obtained from problem  $B$ .

For any  $i \in \{2 \dots N\}$  and any  $j \in \{i + 1 \dots N\}$

$$\begin{aligned} D(a_i, a_j) &= D(b_i, b_j) = l_{b_i b_j}(v_1 \dots v_{n-1}) + 0 \times v_n \\ &= l_{a_i a_j}(v_1 \dots v_n) \end{aligned}$$

And  $l_{a_i a_j}$  is an affine function. This statements is also true for  $i = 1$  and  $j \in \{1 \dots N - 1\}$

Third step, we deduce an affine parameterization of the value associated to edge  $(x, a_{n+1})$  using  $\{v_1 \dots v_n\}$  for any  $x \in \{a_2 \dots a_{n-1}\}$ . For that we use tow tours built as follows:

- $C_1$  is defined as  $(a_1, a_{n+1}, a_n, x, a_{n-1}, a_{n-2}, \dots, a_2)$ ;
- $C_2$  is defined as  $(a_1, a_{n+1}, x, a_n, a_{n-1}, a_{n-2}, \dots, a_2)$ .

Figure 4 presents an example of parameterization of  $D(x, a_{n+1})$ . Tour  $C_1$  evaluated as explain in remark 2. The second tour  $C_2$  have to be evaluated by the oracle.

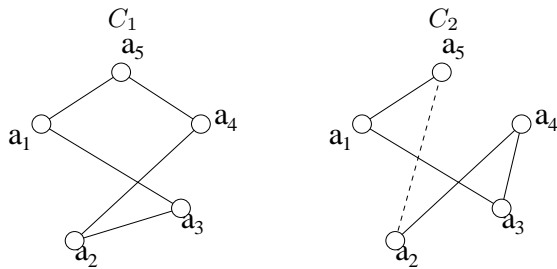


Figure 4: Parameterization of  $D(x, a_{n+1})$ , with  $x = a_2$ .

$$\begin{aligned} k_x &= f(C_1) - f(C_2) \\ k_x &= D(a_{n+1}, a_n) + D(x, a_{n-1}) - D(a_{n+1}, a_1) \\ &\quad - D(a_n, a_{n-1}) \end{aligned}$$

So, we obtain

$$\begin{aligned} D(a_{n+1}, x) &= -k_x + D(a_{n+1}, a_n) + D(x, a_{n-1}) \\ &\quad - D(a_n, a_{n-1}) \\ &= -k_x + D(b_1, b_n) - D(a_{n+1}, a_1) \\ &\quad + D(x, b_{n-1}) - D(b_n, b_{n-1}) \end{aligned}$$

As  $D(x, b_{n-1})$ ,  $D(b_n, b_{n-1})$  and  $D(b_1, b_n)$  are affine expression of  $\{v_1, \dots, v_{n-1}\}$ ,  $D(a_{n+1}, x)$  is an affine expression of  $\{D(a_{n+1}, a_1), v_1, \dots, v_{n-1}\}$

Fourth step, we need to find an affine parameterization of  $D(a_1, a_n)$ . As previously, we build two tours which only one have to be evaluated by the oracle (see figure 5, for an example).

- $C_3$  is defined as  $(a_1, a_{n+1}, a_n, a_2, a_3, \dots, a_{n-1})$ ;
- $C_4$  is defined as  $(a_1, a_n, a_2, a_3, \dots, a_{n-1}, a_{n+1})$ .

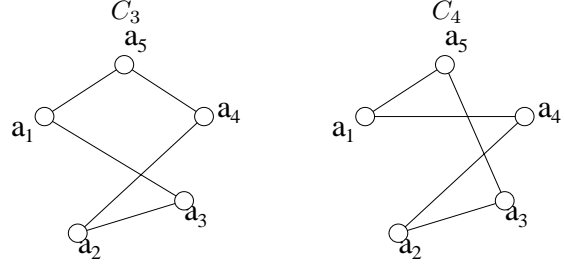


Figure 5: Parameterization of  $D(a_1, a_n)$ .

$$\begin{aligned} k_n &= f(C_3) - f(C_4) \\ k_n &= D(a_{n+1}, a_n) + D(a_1, a_{n-1}) - D(a_1, a_n) \\ &\quad - D(a_{n+1}, a_{n-1}) \end{aligned}$$

So, we obtain

$$\begin{aligned} D(a_1, a_n) &= -k_n + D(a_{n+1}, a_n) + D(a_1, a_{n-1}) \\ &\quad - D(a_{n+1}, a_{n-1}) \\ &= -k_x + D(b_1, b_n) - D(a_{n+1}, a_1) \\ &\quad D(b_1, b_{n-1}) - D(a_{n+1}, a_{n-1}) \end{aligned}$$

As  $D(b_1, b_{n-1})$ ,  $D(b_1, b_n)$  are affine expressions of  $\{v_1, \dots, v_{n-1}\}$ , and  $(D(a_{n+1}, a_{n-1}))$  is an affine expression of  $\{v_1, \dots, v_n\}$   $D(a_1, a_n)$  is an affine expression of  $\{v_1, \dots, v_n\}$ .

### 5.3 End of proof of lemma 1

**Remark 3** In the proof, the variable  $D(a_1, a_{n+1})$  can be seen as the parameter defining every manner to cut of edge  $(b_n, b_1)$  into two parts.

We just prove the first point of property  $P_{N+1}$ , that is we have a parametrization of each edge using  $\{v_1, \dots, v_n\}$ . The second point of property  $P_{N+1}$  is obtained since we have evaluated exactly  $\frac{N-1 \times N}{2}$  tours ( $\frac{(N-2) \times (N-1)}{2}$  for previous step and  $N - 1$  in the current step). That is the minimal number of lines required to span the set of problems equivalent to T.

### 5.4 Corollary

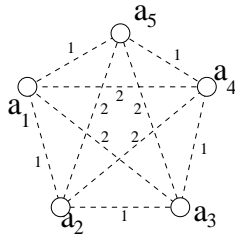
We know a polynomial manner to build a core set for any size of TSP.

## 6 Discussion

In the previous section, we show that TSP is strongly structured if the number of cities is over three. When there is

<sup>3</sup>without additional use of the oracle

only one city, then there is no tour. So, we try to optimize in an empty set. Hence, we cannot say that NFL is false. For problems with two cities, there is only one tour. So, every algorithms perform as well the others<sup>4</sup>. Moreover, such problems do not verify  $P_2$  and they are not structured. In the case of TSP with three cities, there is only one tour too. So, we cannot find a *strict* subset of configurations. So, we do not have the strong structure property. In the same way in case of TSP with four cities, the candidate to be the core set is the whole set of configurations. When the number of cities is greater than or equal to five we have a proper subset. So, these problems are strongly structured. Moreover, when the number of cities is greater than or equal to five, some instances of TSP have configurations out of the core set with different fitness: such an example is presented in figure 6. The greater the problem is the more there are non



This is an non Euclidean TSP. Number near the edge “correspond” to the distance between the nodes.

Figure 6: Example of TSP.

core elements with different fitness. So, NFL does not hold on TSP with size of five or more.

## 7 Conclusion

In this article, we deepen the notion of structure proposed in [6]. We see how to use the notion of structure for classes of problem. Moreover, we see that structures often occurs in combinatorial optimization, even if a core set does not allow explicitly rebuild the data of the problem. In particular, we see that TSP is in this case. In order to be more discriminant, we introduce the notion of strong structure which is verified by GCP. We show that this notion is also verified by TSP.

As an additional result,  $P_N$  states that several TSP problems are equivalent. This result may be a manner to tackle effective or application problems: to solve a problem we could try to solve another equivalent problem on which we remove some constraints (as the Hamiltonian tours constraint for example). As future work, we actually aim at enlarging the set of strongly structured class of problems.

## Bibliography

[1] J. Culberson. On the Futility of Blind Search. Technical Report 96–18, Dept Computer Science, University of Alberta, Edmonton, Alberta, Canada, 1996.

[2] S. Droste, T. Jansen, and I. Wegener. Optimization with randomized search heuristics-the (A)NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, september 2002.

[3] C. Igel and M. Toussaint. On classes of functions for which No Free Lunch results hold. Technical report, Institut für Neuroinformatik, Ruhr-Universität Bochum, ND 04 44780 Bochum - Germany, 2001.

[4] C. Schumacher, M. Vose, and L. Whitley. The No Free Lunch and problem description length. In L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 565–570, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[5] S. Voss and D. Woodruff, editors. *Optimization Software Class Libraries*. Kluwer Academic Publishers, Boston, Hardbound, April 2002.

[6] B. Weinberg and E.-G. Talbi. NFL theorem is unusable on structured classes of problems. In *Congress on Evolutionary Computation (CEC)*, volume 1, pages 220–226, Portland, Oregon, June 2004.

[7] D. Wolpert and W. MacReady. No Free Lunch for optimization. *IEEE Transactions on Evolutionary Computation*, pages 67–82, 1997.

[8] J. Woodward and J. Neil. No free lunch, program induction and combinatorial problems. In C. et al., editor, *Genetic programming*, LNCS 2610, pages 475–484, Essex, UK, 4 2003. Springer-Verlag.

<sup>4</sup>with respect to NFL conclusion