A GRATIS Theorem for Relational Optimization

Mario Köppen, Kaori Yoshida, Kei Ohnishi
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka 820-8502 Japan
Email: mkoeppen@ieee.org, kaori@ai.kyutech.ac.jp, ohnishi@cse.kyutech.ac.jp

***Abstract*—We are studying the NFL Theorems with regard to relational optimization. In relational optimization, we represent the optimization problem by a formal relation, and the solution by the set of maximal (or non-dominated) elements of this relation. This appears to be a natural extension of standard optimization, and covers other notions of optimality as well. It will be shown that in this case, the NFL Theorems do not hold and that there are pairs of algorithms and a performance measure, where one always outperforms the other with regard to this performance measure if averaged over all possible relations. More specifically, the outperforming algorithm is an algorithm, where elements are checked one by one if they are dominated by any other element, while the outperformed algorithm is an algorithm, where elements are checked one by one if they dominate any other element. The proof is accompanied by a complete analysis of a special case, where also other performance measures are shown to be better when using the former algorithm.**

***Keywords*-no free lunch theorems; relational optimization; optimization**

I. Introduction

The No Free Lunch (NFL) Theorems state the equal average performance of search algorithms, if no *a priori* knowledge of the objective function is known, no matter how the algorithm is designed or how the performance is measured from sampled function values [1]. We want to shortly recall the reasoning leading to this consequence. The basic assumption is the existence of a mapping f from a sample space (also called feature space, decision space) X to an objective space Y. In $f : X \to Y$, both X and Y are assumed to be finite sets.

A (deterministic) search algorithm a is set up to sample from X without repetition. In step m the algorithm will sample $x_{(m)} \in X$ and it will learn about the so-far unknown function value $y_{(m)} = f(x_{(m)})$ by consulting the function issuing domain (this has been considered as a *demon action* in past interpretations of the NFL theorems). Based on the new information about the new pair $(x_{(m)}, y_{(m)})$ and also on all former pairs $(x_{(k)}, y_{(k)})$ with $1 \leq k < m$, the algorithm provides a procedure to compute the next sample $x_{(m+1)}$ as long as $m < |X|$. This way, for any f the algorithm will also sample a sequence of elements from Y where sample $y_{(m)} = f(x_{(m)})$. The performance of the algorithm a is only seen in terms of this sampled function value sequence. A typical goal could be to sample a largest value from Y from the unknown f in as few steps as possible[1]. In other words, an algorithm a starts with an unbiased sample $x_{(1)}$ from X (this is fixed for the application of the algorithm to any unknown function). From the "demon" the algorithm receives $y_{(1)} = f(x_{(1)})$. Based on this pair from $X \times Y$ the algorithm decides about the next sample point, $x_{(2)}$, which is different from $x_{(1)}$, receives $y_{(2)} = f(x_{(2)})$, decides on $x_{(3)}$ based on the evaluation of both former pairs and different from $x_{(1)}$ and $x_{(2)}$ etc. The algorithm will stop when the *search space* X is exhausted.

Tables I and II show examples for the sample sequences generated by two different algorithms. In both cases, $X = \{x_1, x_2, x_3\}$ and $Y = \{0, 1\}$. Then, there are $2^3 = 8$ possible function mappings from X to Y. In Tables I and II, the mappings are grouped by so-called atomar *cups* (closed-under-permutation). For each function belonging to a cup, if the elements of X are permuted, the permuted function will also belong to the same cup. For example, function number 5 maps x_1 to 0, x_2 to 1 and x_3 to 1. If we permute the set $\{x_1, x_2, x_3\}$ to $\{x_3, x_1, x_2\}$ while keeping the positional function value assignments (i.e. the first element is mapped to 0, the second and third are mapped to 1) we obtain the function mapping x_1 (now second element) to 1, x_2 (now third element) to 1 and x_3 (now first element) to 0. This corresponds to function number 7 in Tables I and II, which belongs to the same cup. By *atomar cup* we mean that these cups and only these have no true subsets that are also cups, but any cup (including the total set of all possible functions) is a union of atomar cups.

For Algorithm 1, we sample x_2 first, than x_1 and then x_3. This algorithm does not take the corr. y-values of the samples into account when deciding on the next x-value. This is commonly understood as a model of a *random algorithm* (also sometimes called *blind algorithm*). Algorithm 2 samples x_1 first. If $y_{(1)} = f(x_1) = 1$ then x_2 will be sampled next, x_3 otherwise. For the third step, there is only one x-value remaining (either x_3 or x_2). Table I shows the sampled function value sequences of Algorithm 1 for all possible functions $f : X \to Y$, and Table II the same for Algorithm 2.

[1]This is a deterministic algorithm, as sampling according to a probability distribution is not considered here. However, it has already been shown in [1] that the effects of sampling from distributions just unfold over all possible deterministic states.

Table I
SAMPLING OF ALL BINARY FUNCTIONS OF 3 ARGUMENTS BY ALGORITHM 1 ($x_2 \rightarrow x_1 \rightarrow x_3$). THE FUNCTIONS ARE GROUPED BY CUPS.

#	$f(x_1)$	$f(x_2)$	$f(x_3)$	$y_{(1)}$	$y_{(2)}$	$y_{(3)}$
1	0	0	0	0	0	0
2	0	0	1	0	0	1
3	0	1	0	1	0	0
4	1	0	0	0	1	0
5	0	1	1	1	0	1
6	1	0	1	0	1	1
7	1	1	0	1	1	0
8	1	1	1	1	1	1

Table II
SAMPLING OF ALL BINARY FUNCTIONS OF 3 ARGUMENTS BY ALGORITHM 2 (x_1 FIRST, IF $f(x_1) = 1$ THEN x_2 NEXT ELSE x_3, AND LAST THE REMAINING x SAMPLE). THE FUNCTIONS ARE GROUPED BY CUPS.

#	$f(x_1)$	$f(x_2)$	$f(x_3)$	$y_{(1)}$	$y_{(2)}$	$y_{(3)}$
1	0	0	0	0	0	0
2	0	0	1	0	1	0
3	0	1	0	0	0	1
4	1	0	0	1	0	0
5	0	1	1	0	1	1
6	1	0	1	1	0	1
7	1	1	0	1	1	0
8	1	1	1	1	1	1

The key for understanding the NFL Theorem is within the cups. It is obvious that the sampling sequence of a function is a permutation of the ordered set of function values. For example, in Table I the ordered set of function values for function number 4 is $\{1, 0, 0\}$ and the sequence of sampled function values is $\{0, 1, 0\}$. But also, if an algorithm samples two different functions from the same cup than the corresponding function value sequences need to be different. Assume the opposite: there are functions f_1 and f_2 for which the sequences of sampled Y values are the same. Since the algorithm always starts with the same $x_{(1)}$ and the sampled Y values are the same, we find that $f_1(x_{(1)}) = f_2(x_{(1)})$. Then, also the decision for the next sample $x_{(2)}$ is based on the same conditions, the knowledge of the pair $(x_{(1)}, y_{(1)})$ and will be the same for both functions. Then, also $y_{(2)} = f_1(x_{(2)}) = f_2(x_{(2)})$, the algorithm will decide on the same next sample $x_{(3)}$ for both functions etc. Thus after finishing the search it will come out that f_1 and f_2 have the same values for all $x_{(i)}$ and therefore also for all x_i.

Thus, all sample sequences of Y values, listed for all functions of the same cup, must be different from each other. Then their list is a permutation of the list of ordered function value sequences for functions from the same cup. For example, the three functions 5, 6 and 7 in Table I are sampled as follows: function 5 by the ordered set of function values of function 6 (which is $\{1, 0, 1\}$), function 6 by the one of function 5, and function 7 by its own ordered set. In the same sense, the function set $(2, 3, 4)$ is sampled as the function set $(2, 4, 3)$ by Algorithm 1, and as the function set $(3, 2, 4)$ by Algorithm 2.

Now, the NFL Theorem here is a direct consequence of this property: if we evaluate algorithms by means of their value sampling sequences for all functions of a cup alone, the evaluation will always be based on the same sampling sequences, just in a different order. For example, if we count the average number of 1s found after two steps, it will be $8/8 = 1$ for Algorithm 1 and also for Algorithm 2, since the evaluation is only based on the number of 1s in the sequences 00 (cup with function 1), 00, 01, 10 (cup with functions 2, 3 and 4), 01, 10, 11 (cup with functions 5, 6 and 7) and 11 (cup with function 8), only appearing in a different order.

Up to this point, we have said nothing new about the NFL Theorems, but we will need the arguments for the following discussion. Basically, the NFL Theorems confirm that a performance evaluation based on sample functions alone is ill-posed. If the evaluation will not put any specific demands on the sample function, it will include functions that are far beyond any practical concern, but "average out" performance gains to any wanted degree. Therefore, the focus of past studies has been on conditions, for which the NFL Theorems do not hold. Several of them have already been identified. For example, the proof requires all functions to be feasible, a condition that is often not fulfilled in combinatorial optimization problems[2]. Also, in the presented form, the theorem is also sometimes referred to as Generalized NFL-Theorem: it will not hold for a subset of functions that is not cup, i.e. closed under permutation [2].

In [3] and [4] it was shown independently that the NFL Theorems are also valid for multi-objective optimization, where the elements of Y are vectors from R_n. However, in [5] it was shown that an often used component of evolutionary multi-objective optimization algorithms, the archive, gives raise to a condition where the NFL Theorems will not hold. The issue is that archiving strategies break the independence of the measurement from the sampled elements. In the arguments given above, the performance of an algorithm applied to an unknown function is derived from the sampled sequence of values from the set Y alone. However, an archive keeps "good" sample points from X for some time, but if it grows too large, there is a strategy to remove a number of points. This is usually an evaluation of X-values alone (e.g. the so-called crowding measures), but it will affect the sampled sequence of Y values as well. The authors gave the name GRATIS theorems to such kind of differences in average performance assessments, to indicate

[2]Take the Traveling Salesperson Problem with 5 cities, it happens that the opposite of a route, i.e. the selection of all connections that are not included in the route is also a route. Thus, there are linear dependencies among the route costs, and a "free" assignment of costs to routes is not possible (*Christian Igel, personal communication*).

the existence of a "free lunch."

In this paper, we will expand this analysis by considering relations among elements of the set X. It should be noted that the performance measure used in the NFL Theorems is completely arbitrary, and not necessarily related to any concept of "maximality." For a binary space Y, we may also count the number of cases that a function value 1 was sampled after having sampled two times the function value 0 in the preceding two steps. Such measures do not express any concept of optimality, but are nevertheless covered by the NFL Theorems. Therefore, we want to study the case where the initial idea of optimization is directly represented by the performance measure, and consider therefore relational optimization. In this case, relations formally represent the notion of optimality, where the maximal, or non-dominated elements represent the seeked optimal set. We find such relations among the elements of X, if we compare them via the corresponding function values in Y. In fact, querying the function for a function value of a sample x or querying whether an unknown function value of x is in a kind of better-relation to another one appears to be a similar compensatory action for incomplete knowledge, while the latter is of direct relevance for the goal of optimization. We will clarify the notion of relational optimization and how hybrid meta-heuristics can be used to obtain this goal in the next section. Then, it can be shown that the NFL Theorems do not hold for this kind of optimization, and that in fact some algorithms have a steady better average performance over all possible relations (and not all possible functions) than other algorithms. The proof will be given in Section III, followed by a demonstration example in Section IV.

II. Relational Optimization

Given a feasible space X, a (binary) relation R is a subset of $X \times X$. Thus, for two points x_1 and x_2 from X it is said that x_1 is in relation R to x_2 if the ordered pair (x_1, x_2) belongs to this subset. Relations may represent various things like equality, similarity, dependency etc. depending on the nature of X, and they can be represented in various equivalent forms (list of ordered pairs, tabular form, binary matrix, directed graph, logical property etc.). Here, we are focusing on the interpretation of a relation as a larger relation, and will write $>_R$ to represent that x_1 is of better quality, or preferred to x_2 in some sense.

For each relation (no matter if it represents such betterness) we may assign two characteristic subsets of X [6].

Definition 1. *Given a relation $>_R$ then the **best set** is defined as the set of all x' such that for any other $x \in X$ $x' >_R x$ holds.*

Definition 2. *Given a relation $>_R$ then the **maximum set** is defined as the set of all x' such that for no other $x \in X$ $x >_R x'$ holds.*

The best set is sometimes called set of greatest elements, and the maximum set is also called set of maximal elements or non-dominated set. Also note that both definitions are dual, the best set is the maximum set of the converce complement relation to $>_R$ and vice versa.

Once having specified a relation $>_R$ the goal of *relational optimization* is to find its maximum set (also sometimes the best set). Thus, the relation provides a specific notion of optimality. We can find this kind of optimization in various circumstances:

"Multivariate optimization": it appears to be a generalization of the standard optimization task, where a function $f : X \subseteq R^n \rightarrow Y \subseteq R$ is given (usually called fitness function, quality function, or cost function) and the goal is to find a $x' \in X$ such that $f(x')$ is maximal[3]. With regard to the domain of f (i.e. a subset of R) this appears to employ the standard larger relation between real numbers. However, with regard to the feasible space X this refers to probing an pre-order relation between elements of X: $x_1 \geq_f x_2$ if and only if $f(x_1) \geq f(x_2)$. Then, the goal is to find the best set of $>_f$. In this case, best set and maximum set coincide so the definitions are equivalent.

Pareto front: In multi-objective optimization (MOP), the Pareto-dominance vector relation is used where for $x_1, x_2 \in R^n$ $x_1 >_P x_2$ holds if and only if each component of $f(x_1)$ is larger than or equal to the corr. component of $f(x_2)$ and at least one component is larger (here, f is a mapping from a feature space $X \subseteq R^n$ into an objective space $Y \subseteq R^m$). Then, the goal of a MOP is specified as finding the non-dominated elements of X, i.e. the maximum set of the Pareto-dominance relation.

Fair distribution and social choice: in mathematical economy, problems are studied where a limited resource (for example indivisible goods) have to be distributed to a number of agents. Concepts like Pareto-efficiency and envy-freeness of such a distribution are represented by relations or sets of preference relations assigned to the agents (fair division problem [7], preference-based search and notions of B-preference, E-preference etc. [8]). Suzumura [9] introduced relations to represent social choice (as given for example by a voting scheme): a social choice is *rationalizable* if there is a relation such that the social choice corresponds with the best set of the relation.

Fairness relations: similar to mathematical economy, in data communication also relations are used to represent fair sharing of resources, but the relations are much more specific here. Among such relations, we can find maxmin fairness [10] and proportional fairness [11]. For example, maxmin fairness is defined as follows:

Definition 3. *Given a feasible space $X \subseteq R_n$. For two different elements (vectors) x_1 and x_2 from X it is said that x_1 **maxmin fair dominates** x_2 ($x_1 >_{mmf} x_2$) iff for each*

[3]Of course, minimal if we consider f as a cost function.

i with $x_{2i} > x_{1i}$ there is at least one $j \neq i$ such that (1) $x_{1i} \geq x_{1j}$ and (2) $x_{1j} > x_{2j}$.

In a convex feasible space, maximum and best sets of this relation coincide and define the maxmin fair state that can be used to assign, for example, fair traffic rates in a data communication network.

Interactive Genetic Algorithms: in interactive evolutionary computation, the fitness function is not given explicitely but in terms of comparisons by a user. This refers to sampling an unknown relation between the elements of the design space (note that it probably needs the concept of n-ary relations as subsets of X^n as it cannot be ensured that user preferences can always be represented by a binary relation) and employ it to approach the maximum set of this unknown relation.

All these examples show that in fact, relational optimization is a common approach in many fields and allows for much more flexibility in the specification of optimization goals. It also allows to specify such optimization problems essentially as the search for the maximum set of an arbitrary relation. One concern might be raised about empty maximum sets. In fact, they exist, and would render the corresponding relational optimization problem as one without a solution. But two arguments are in place: if we consider the standard optimization, Pareto front, or most of the fairness relations, the maximum sets can be shown to be non-empty. In other cases, the problem might be also related to the propper specification of a relation such that the maximum sets are meaningful, i.e they are not empty, but also much smaller than the feasible space. This aspect is of higher interest for the social choice and distribution theme, where existence of a corresponding scheme (i.e that the goals of the various agents are fulfillable) is of higher importance.

Last but not least, we also have to comment then on how these maximum sets can be found at all. It appears that also here, meta-heuristic approaches like evolutionary computation, swarm intelligence, tabu search and simulated annealing can provide valuable search algorithms. The basic experience comes from the transition from (single-objective) evolutionary computation and swarm intelligence to their multi-objective "counterparts" by replacing internal computations and design principles basing numerical comparisons with the Pareto-dominance relation. This gives the family of so-called Evolutionary Multi-Objective Optimization Algorithms (EMOA). It has already been demonstrated that an equivalent transition can be done with some fairness relations [12]. For example, in [13] the SPEA2 algorithm [14] is extended to a generalized SPEA2 in order to approximate the maxmimum set of various fairness relations in the problem domain of wireless channel allocation.

III. A GRATIS Theorem for Relational Optimization

We provide a model for relational optimization by deterministic search, which is similar to the deterministic search for function optimization, leading to the NFL Theorems in the latter case. Again, consider a set X and a relation $>_R$. The searchspace here is the set of pairs from $X \times X$ and for each pair we use a function assignment 1 if the pair belongs to the relation, and 0 if not. We consider the performance of a over all possible relations (there are $2^{|X|}$ of them). Then, we sample pair by pair in a manner prescribed by a search algorithm a in order to find the maximum set of the unknown relation. For simplicity, we exclude pairs of equal elements from the search, since they are not of interest with regard to maximum sets (reducing the searchspace to $2^{|X|(|X|-1)}$ elements). During the sampling, the algorithm finds about elements of X belonging to the relation, based on two rules:
Rule 1. If for some x' the algorithm has sampled all pairs (x, x') and the function assignment was 0 in all cases, it is confirmed that x' belongs to the maximum set.
Rule 2. If for some sampling $(x, x') \rightarrow 1$ (i.e. $x >_R x'$) it is confirmed that x' does not belong to the maximum set. Further pairs containing x' at the second position will not be sampled anymore.

The algorithm starts with a predetermined sample pair $p_1 = (x_1^1, x_2^1)$. Based on whether the value 0 or 1 is unrevealed, i.e. whether the first point of the first sample pair is in relation to the second point of the first sample pair, the algorithm decides on the next sample pair $p_2 = (x_1^2, x_2^2)$. Then, based on the results for p_1 and p_2 the algorithm continues with a selection of p_3 etc. This model corresponds with the model for function optimization, where X is the set of all pairs with different elements and $Y = \{0, 1\}$. The only difference is that the search will not include elements anymore that have been confirmed to not belong to the maximum set according to Rule 2 in a former step of the algorithm.

The performance of a search algorithm a for such a relational search problem is based on the number of elements that have been confirmed to belong to, or excluded after a number m of algorithm steps. For example, we can simply count how many elements of the maximum set have been confirmed after m steps.

For this kind of relational search, we will show the following Theorem.

Theorem 1. *There exist algorithms a and b and a performance measure c such that for any relational search problem for some number of steps algorithm a always outperforms algorithm b in average over all possible relations.*

Proof: We take some fixed order of the n elements of X, yielding $X = \{x_{(1)}, x_{(2)}, \ldots, x_{(n)}\}$ and consider two algorithms $exploit$ and $explore$, working as follows.

Algortihm $exploit$ first tests all pairs $(x_{(i)}, x_{(1)})$ with $i = 2, \ldots, n$, then all pairs $(x_{(i)}, x_{(2)})$ with $i = 1, 3, \ldots, n$ etc. In the comparison, we exclude elements that have already been confirmed to not belong to the maximum set according to Rule 2. Thus, algorithm $exploit$ exhaustively tests elements of X one by one if they belong to the maximum set or not. Algorithm $explore$ does the opposite. First, it tests all pairs $(x_{(1)}, x_{(i)})$ with $i = 2, \ldots, n$, then all pairs $(x_{(2)}, x_{(i)})$ with $i = 1, 3, \ldots, n$ etc, and also skipping pairs according to Rule 2. Thus, algorithm $explore$ tests elements of X one by one whether they are dominating any other element of X or not.

We choose as performance measure c the number of elements of the maximum set that have been found after $(n-1)$ steps. Then, algorithm $exploit$ has surely confirmed whether $x_{(1)}$ belongs to the maximum set of the sampled relation $>_R$ or not. Since there is always at least one relation whose maximum set contains $x_{(1)}$, the average performance over all possible relations is larger than 0. On the other hand, algorithm $explore$ cannot confirm even a single element of the maximum set since it will never test any element of X against all other. At most, it can confirm $(n-1)$ elements NOT belonging to the maximum set, and also, Rule 2 will never apply since no second element of the sampled pairs appears twice in the first $(n-1)$ steps. Thus, the average performance over all possible relations is 0. As a consequence, for this measure and $(n-1)$ steps, algorithm $exploit$ will always outperform algorithm $explore$. ■

IV. Demonstration

In this section, we want to illustrate the result of the former section by a complete example evaluation. We take as an example the set $X = \{a, b, c\}$. Thus, the searchspace for the relational search is composed of the pairs $\{(a,b),(a,c),(b,a),(b,c),(c,a),(c,b)\}$ (again, we do not consider equal pairs). Then, the algorithm $exploit$ samples them in the order $(b,a),(c,a),(a,b),(c,b),(a,c),(b,c)$ i.e. first checks if any element dominates a, then b and then c. After at most 2 steps $exploit$ can decide whether a belongs to the maximum set, after at most 4 steps for b and at most 6 steps for c. The algorithm $explore$ samples in the order $(a,b),(a,c),(b,a),(b,c),(c,a),(c,b)$ and can decide about a membership to the maximum set after at most 5 steps, b 6 steps and c 4 steps.

For space reason, we cannot present the evaluation for all possible $2^6 = 64$ possible relations. To illustrate the different action of both algorithms, we take the relation $R = \{(a,b),(a,c),(b,c),(c,b)\}$. The maximum set of R contains a only as the only non-dominated element. Algorithm $exploit$ first tests (b,a). This does not belong to R so nothing can be concluded. Then the algorithm tests (c,a) which is also not contained. We learn that a belongs to the maximum set, since neither b nor c are in relation to it. Next is (a,b) which belongs to R and we learn that b is dominated by a and cannot belong to the maximum set. Therefore, there is no need to test the next pair (c,b) in the initial sequence, and the algorithm continues with (a,c). Also this is contained in R and c does not belong to the maximum set. The algorithm can stop after step 4, since all three element membership to the maximum set have been decided.

Algorithm $explore$ starts with (a,b) and since this is in R we learn that b is not an maximal element. The second step (a,c) reveals that also c is not maximal. The third step (b,a) does not provide any information, and the following (b,c) can be skipped, since c has already been ruled out. The last step (c,a) then shows that a is neither dominated by b or c, and the maximum set is found after 4 steps as well.

This processing will be performed for all 64 possible relations, and we may present a few results for this case. We also compared with an algorithm $mixed$ that samples (a,b) first, and if it is contained in R follows the order $(a,c),(b,a),(c,a),(b,c)$ ((c,b) is skipped by Rule 2), otherwise the order $(b,a),(c,a),(b,c),(a,c),(c,b)$ (probably shorter if Rule 2 applies). Among all relations, 1 has a maximum set of size 3, 9 of size 2, 27 of size 1, and in 27 cases, the maximum set is empty.

Table III
Average number of elements for the 3 algorithms with confirmed membership or non-membership to the maximum set, in fractions of 64.

alg	1	2	3	4	5	6
$exploit$	32	80	120	160	184	192
$explore$	32	64	96	152	184	192
$mixed$	32	64	104	152	184	192

Table III shows the average number of confirmed memberships or non-memberships for increasing number of steps. Also here, $exploit$ clearly outperforms $explore$. Despite the fact that algorithm $explore$ is designed to make rather early decisions about non-membership to the maximum set, it lacks behind algorithm $exploit$ and also $mixed$ (thus, the naming "explore" might actually not be appropriate). Moreover, for the 37 relations with one or more elements in the maximum set, algorithm $exploit$ needs an average 3.1 steps to confirm the first element, algorithm $explore$ 4.3 steps and algorithm $mixed$ 4.0 steps. For the 10 relations with at least 2 elements, algorithm $exploit$ needs 4.8 steps to find a second element of the maximum set, $explore$ needs 5.2 steps and $mixed$ 5.1 steps. For the only case that the maximum set has 3 elements (the empty relation) all algorithms need 6 steps to confirm the 3rd element. In each case, algorithm $exploit$ finds elements of the maximum set fastest, followed by $mixed$ and slowest is $explore$. However, in average all algorithms finish after 4.5 steps.

V. Outlook

Relations play a fundamental role in the design of models of the real world, and only by means of a relation, we can establish efficient notions for optimality. In a deterministic search algorithm, seeking to maximize a function $y = f(x)$, both aspects seem to be equivalent: querying an unknown function value for a sample x from the searchspace and then comparing the numerical result to other already sampled function values, or querying directly whether this sample is in a pre-order relation via the function values to other already sampled elements of the searchspace. We considered this latter aspect of optimization as relational optimization, and have provided various contexts where this modality of optimization appears. Then, it was shown that, in contrary to the first aspect and with regard to all possible relations (and not all possible functions), outperforming algorithms exists. More specifically, an algorithm that tests elements one by one, whether there is any other dominating element, seems to outperform an algorithm that tests elements one by one, whether they are dominating any other element, in various cases. It is a subject of further investigation if this is generally the case (not only in the case used for the proof of the GRATIS theorem that was presented in this paper).

As a "standard" argument against the NFL Theorems, that the vast majority of considered functions is of no practical relevance, it might also be brought up against the present analysis. However, we adopt the point of view that the NFL Theorems is about the fact that a performance evaluation based on unspecified functions is ill-posed, as this automatically covers all these "useless" functions as well. The same holds for relational optimization, but here is the result that we can provide such a means of performance, and do not need to care about the practical importance of appearing relations. However, we may have to fix the boundary conditions, under which all possible relations actually can appear during the functional mapping involved in an optimization task. For example, if just mapping into the set of real numbers, relations among the elements of X also have to be complete and transitive, so a relevant number of possible relations is excluded. In the same line of arguments, it has also to be considered whether specific properties of relations, especially transitivity, will have any impact on the expected performance of algorithms. Last but not least, the analysis here has been restricted to binary relations and a corrsponding study of n-ary relations might become more interesting in this regard, as concise approaches to these extended relations have not been much investigated so far.

References

[1] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1995.

[2] C. Schumacher, M. D. Vose, and L. D. Whitley, "The no free lunch and problem description length," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001, pp. 565–570.

[3] M. Köppen, "On the benchmarking of multiobjective optimization algorithm," in *Knowledge-Based Intelligent Information and Engineering Systems (KES 2003), Proceedings*, ser. LNAI 2773, V. Palade, R. J. Howlett, and L. Jain, Eds. Springer-Verlag Heidelberg, 2003, pp. 379–385.

[4] D. Corne and J. Knowles, "No free lunch and free leftovers theorems for multiobjective optimization problems," in *Evolutionary Multi-Criterion Optimization (EMO 2003) Second International Conference, Faro, Portugal, April 2003, Proceedings*. Springer LNCS, 2003, pp. 327–341.

[5] ——, "Some multiobjective optimizers are better than others," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 4, 2003, pp. 2506–2512.

[6] A. K. Sen, *Collective Choice and Social Welfare*. Holden-Day, San Francisco, 1970.

[7] S. Bouveret and J. Lang, "Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, 2005, pp. 935–940.

[8] U. Junker, "Preference-Based Search and Multi-Criteria Optimization," *Annals of Operations Research*, vol. 130, no. 1-4, pp. 75–115, 2004.

[9] K. Suzumura, *Rational Choice, Collective Decisions, and Social Welfare*. Cambridge University Press, 2009.

[10] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, N: Prentice Hall, 1992.

[11] F. Kelly, "Charging and rate control for elastic traffic," *Eur. Trans. Telecomm.*, vol. 8, pp. 33–37, Jan./Feb. 1997.

[12] M. Köppen, R. Verschae, K. Yoshida, and M. Tsuru, "Comparison of evolutionary multi-objective optimization algorithms for the utilization of fairness in network control," in *Proc. 2010 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010), Istanbul, Turkey*, 2010, pp. 2647 –2655.

[13] M. Köppen, K. Yoshida, and K. Ohnishi, "Meta-heuristic optimization reloaded," in *2011 Third World Congress on Nature and Biologically Inspired Computing, Proc.*, Salamanca, Spain, October 2011, pp. 569–575.

[14] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," in *EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., Athens, Greece, 2002, pp. 95–100.