Statistical methods for the comparison of stochastic optimizers

Marco Chiarandini* Dario Basso† Thomas Stützle*

*Fachgebiet Intellektik, Fachbereich Informatik, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt
{machud,stuetzle}@intellektik.informatik.tu-darmstadt.de

†Department of Statistics, University of Padova
Via Cesare Battisti 241, 35100 Padova.
dario@stat.unipd.it

1 Introduction

The comparison of stochastic algorithms for optimization requires the use of appropriate statistical methods for the analysis of results. Commonly used *parametric tests*, like the t-test or ANOVA procedures in experimental design, are based on the assumption that algorithms' results are normally distributed. However, this assumption is often violated because the distribution of the cost values of the final solutions is commonly very asymmetric. For example, in the case of minimization problems the distribution is likely to be bounded on the left and skewed to the right.

The alternative to parametric statistical methods are non-parametric methods such as *rank-based tests* and *permutation tests*. These tests typically rely on less strong assumptions than parametric tests and especially refrain from the assumption of normality. While rank-based tests are widely known and applied, permutation tests appear to have been ignored so far and we are the first to present their application in the analysis of experimental designs on stochastic optimizers. We focus on the analysis of *complete block designs*, where the algorithms constitute the treatment factor and instances the blocking (or nuisance) factor. We consider two designs: *running each algorithm once on various instances* and *running each algorithm several times on various instances*. For the same number of experiments the first setting guarantees that the variance of the resulting average performance estimate is minimized [1]. Nevertheless, the second setting is commonly used in many publications and is specially relevant if relatively few benchmark instances are available. We do not consider the case of running algorithms several times on only one instance, because this setting is not relevant for generalizing the performance of algorithms in practice.

2 Statistical methods

In presence of results on several instances the results must be normalized because different instances may imply differ scales. A common transformation is to convert the result $c(i)$ on instance i into the relative deviation from the optimal solution $c_{min}(i)$, *i.e.*, $e_1(c,i) = (c(i) - c_{min}(i))/c_{min}(i)$. Alternatively, a measure that guarantees invariance under some trivial transformations is $e_2(c,i) = \frac{c(i)-c_{min}(i)}{c_{max}(i)-c_{min}(i)}$, where $c_{max}(i)$ is the worst solution for instance i [12]. If c_{min} and c_{max} cannot be computed efficiently, which is typically the case for $\mathcal{NP}$-hard problems, surrogate measures must be used like the best known solution value and the value returned by some simple heuristic, respectively. Differently, for the application of rank-based tests, data are transformed into *ranks within* each instance. This removes the need for normalization but necessarily reduces the amount of information, because it neglects the entity of the differences between algorithms on the instances.

Statistical tests are used to determine whether the observed differences are real or are due to chance. If the "parametric" distribution assumptions are valid, then the distribution of some test statistics may be derived theoretically. However, if all the parametric assumptions are not satisfied, it is still possible to derive the distribution of a test statistic from the original data by the use of *permutation tests*. A permutation test of hypotheses works as follows: Firstly, a statistic S is chosen and its value $S_0 = S(X)$ is computed from the original set of observations X. Secondly, the permutation distribution of S is constructed by generating all possible rearrangements (permutations) of the observed data among the algorithms and computing the value of the test statistic S on each rearrangement X^*: $S^* = S(X^*)$ (rearrangements correspond to exchanging a number of observations between the treatment that are compared). Thirdly, the upper α-percentage point z of the distribution of S^* is derived (in case of one-sided tests) and the null hypothesis (H_0) of no difference between treatments is accepted or rejected depending on whether S_0, measured on the original observations, is smaller or larger than the z value. If the sample size is not very small, generating all possible permutations at the second step becomes computationally prohibitive; in this case, an approximate permutation distribution can be obtained by a re-sampling procedure without replacement from the space of all permutations. With a sample of size B, the value z in the third step is $\#\{S^* \geq z\}/B = \alpha$ (for one-sided test). *Rank-based tests* are simply permutation tests applied to the ranks of the observations rather than their original values. Since for a given sample size, ranks have the same values, the critical values of the test statistics can be tabulated and heavy computations be avoided [6].

Design 1: One single run on various instances. The data consist of b mutually independent k-variate random variables $X = (X_{h1}, X_{h2}, \ldots, X_{hk})$, with $h = \{1, \ldots, b\}$, b the number of instances (*i.e.*, the blocks) and k the number of algorithms (*i.e.*, treatments). The random variable X_{hi} describes the observation relative to the i-th algorithm on instance h. The data can be arranged in a matrix as shown in Figure 1.

The question of interest is whether the algorithms behave differently. To test this, each measured response X_{hi} is expressed as the linear sum $\tau + \mu_i + \theta_h + \epsilon_{hi}$, where τ is the "true response", ϵ_{hi} is an error given by the presence of nuisance variation, μ_i are the algorithm effects, and θ_h are the instance effects (we assume, *a priori*, that algorithms may behave differently on different instances). The variance of the model is then decomposed in *between-group* variance and *within-group* variance. The analysis of whether algorithms rather than

	Algorithm 1	Algorithm 2	...	Algorithm k
Instance 1	X_{11}	X_{12}		X_{1k}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
Instance b	X_{b1}	X_{b2}		X_{bk}

Figure 1: Design with several algorithms on various instances and one single measurement.

random variations are the main source of variance is based on the statistic S defined as:

$$S = \frac{\text{MSA}}{\text{MSE}}; \quad \text{MSA} = \frac{b \sum_{i=1}^{k} (\bar{X}_{.i} - \bar{X}_{..})^2}{k-1}; \quad \text{MSE} = \frac{\sum_{h=1}^{b} \sum_{i=1}^{k} (X_{hi} - \bar{X}_{h.} - \bar{X}_{.i} + \bar{X}_{..})^2}{bk - b - k + 1} \tag{1}$$

where $\bar{X}_{.i} = \sum_{h=1}^{b} X_{hi}/b$, $\bar{X}_{h.} = \sum_{i=1}^{k} X_{hi}/k$ and $\bar{X}_{..} = \sum_{i=1}^{k} \sum_{h=1}^{b} X_{hi}/bk$ [4]. Under the assumptions of *independence*, *homoschedasticity* and *normality* of the sampled distributions X, the ratio S follows a Fisher distribution with degrees of freedom $k-1$ and $bk-b-k+1$; this corresponds to the well known *parametric analysis of variance*, ANOVA [4, 9]. If, even after some transformation of data, the assumption of normality remains violated, we may want to restrict our assumptions to let X be independent and identically distributed (hence, still homoschedastic but not anymore normally distributed). If H_0 states that all algorithms behave the same on a given instance, we can consider data exchangeable "within" instances (not "between" instances). In other terms, under H_0 all $(k!)^b$ permutations of data, obtained by the rearrangement of the k observations per instance, are equally likely. The *permutation test* uses the test statistic of Equation 1 or the equivalent version $S' = \sum_{i=1}^{k} (\sum_{h=1}^{b} X_{hi})^2$ [5]. If also the assumption of homoschedasticity is not verified, the transformation of data into ranks within each block removes the effect of outliers and rank-based test appear to be safer against error. In that latter case, the *Friedman test* is the appropriate *rank-based test* [3, 6].

Once the null hypothesis that all algorithms perform the same has been rejected, a researcher is typically interested in statistically examining which algorithms perform significantly better than others. In an *all-pairwise comparison*, each algorithm is compared to every other algorithm which entail a family of $c = k(k-1)/2$ tests. Multiple comparisons can be carried out by computing *simultaneous confidence intervals* [7]. Confidence intervals are determined for the differences between the observed means $\bar{X}_{.i}$ and $\bar{X}_{.j}$ of any pair of treatments i and j with mean response μ_i and μ_j. Then, the differences are declared significant if $|\bar{X}_{.i} - \bar{X}_{.j}| >$ MSD, where MSD is the *minimum significant difference* derived from the corresponding confidence interval. Well known parametric methods for computing MSDs are, for example, *Scheffé's method* or the more powerful *Tukey's Honest Significant Difference method* [7, 4]. An algorithm for the permutation approach for computing MSDs between two treatments i and j is given under the Design 2, in Algorithm 2, and can be easily adapted to the current design by removing the third index in all the X terms. The algorithm extends the procedure Compute_pairwise_MSD, described in [10], for computing confidence intervals for mean differences between two treatments to the case of c comparisons. In order to maintain the probability that all c confidence intervals contain the corresponding parameter at the defined level of confidence $1-\alpha$ and thus to guarantee that the rate of making incorrect consequent decisions remains the nominal one, the procedure Compute_pairwise_MSD is repeatedly ap-

	Algorithm 1	**Algorithm 2**	...	**Algorithm k**
Instance 1	$X_{111}, \ldots, X_{11r}$	$X_{121}, \ldots, X_{12r}$		$X_{1k1}, \ldots, X_{1kr}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
Instance b	$X_{b11}, \ldots, X_{b1r}$	$X_{b21}, \ldots, X_{b2r}$		$X_{bk1}, \ldots, X_{bkr}$

Figure 2: Design with several algorithms on various instances and repeated measures.

plied until an interval width is found that satisfies all c pairwise comparisons simultaneously. A rank-based method for computing MSDs can be derived from the *Friedman test* [3]. In this case, the results of all algorithms are ranked jointly within each instance, and differences of any two algorithms i and j are declared significant if $|\bar{R}_i - \bar{R}_j| > \text{MSD}$, where $\bar{R}_l$ is the average rank of algorithm l. Alternatively, the *Wilcoxon matched-pairs signed-ranks* test with Holm's adjustment of the comparison-wise α-level [11, 3] can be used on each pair of algorithms. In [7] this method is said to be preferable over the Friedman test.

The graphical representation of simultaneous confidence intervals used in the parametric case may be extended also to the other two cases. The plot is obtained by attaching error bars derived by the MSD values to a scatter plot of the estimated effects versus algorithm labels. The length of the error bars are adjusted so that the population means of a pair of treatments can be inferred to be different if their bars do not overlap, *i.e.*, $\bar{X}_i \pm \text{MSD}/2$. This representation is limited to the case of balanced designs which, however, should always be obtainable in experiments on algorithms. We give an example of such a plot in Section 3.

Design 2: Several runs on various instances For each pair algorithm–instance r independent *runs* are available, $r > 1$. Again, the instances are blocks and observations in different blocks are assumed to be independent. Data may be viewed as random samples of size r of b mutually independent k-variate random variables, $X = (X_{hi1}, X_{hi2}, \ldots, X_{hir})$, where k is the number of treatments, b the number of blocks, and r the number of observations per experimental unit. The random variable X_{hit} is the t-th realization on block h for algorithm i. Data may be visualized as in Figure 2.

Two cases may be distinguished. The case in which the interaction between individual instances and algorithms is considered unimportant and the case in which algorithms may rank differently on different instances and therefore an interaction term $(\theta\alpha)_{hi}$ must be included in the linear model. This gives rise to two slightly different ratio statistics based on Equation 1 with, respectively,

$$\text{MSE} = \frac{\sum_{h=1}^{b}\sum_{i=1}^{k}\sum_{t=1}^{r}(X_{hit} - \bar{X}_{h..} - \bar{X}_{.i.} + \bar{X}_{...})^2}{bkr - b - k + 1} \quad \text{or} \quad \text{MSE} = \frac{\sum_{h=1}^{b}\sum_{i=1}^{k}\sum_{t=1}^{r}(X_{iht} - \bar{X}_{jh.})^2}{bk(r-1)} \tag{2}$$

Again, under the parametric assumptions the statistic S from Equation 1 follows the Fisher distribution with degree of freedom $k - 1$ and the values at the denominators of MSE from Equation 2. Permutation tests for this design require the use of *synchronized permutations* [10]. An intermediate statistic is computed within each instance h as $S_{ij|h} = \sum_t X_{hit} - \sum_t X_{hjt}$.

```
Procedure Compute_statistic();
S* = 0; generate a random permutation π of labels {1, ..., 2r};
for all k(k − 1)/2 distinct pairs (i, j) do
    for h = 1, ..., b do
        Let X_pool = X_hi ∪ X_hj be the set of the 2r pooled observations;
        X*_hi = X_pool[π(1), ..., π(r)];
        X*_hj = X_pool[π(r + 1), ..., π(2r)];
        S* = S* + (Σ_t X*_iht − Σ_t X*_jht)^2;
    end
end
```

Algorithm 1: An algorithm for synchronized permutations in a two factors design.

Then, for each instance there are $k(k-1)/2$ intermediate statistics, each comparing two different algorithms. The final statistic is $S = \sum_{i<j} \left(\sum_{h=1}^{b} S_{ij|h} \right)^2$, that, with synchronized permutations, is uncorrelated from the effect of interaction. Each S^* for deriving the distribution of S can be implemented as indicated by Algorithm 1. The use of the same permutation π ensures that the number of observations swapped between algorithms i and j is maintained synchronized among the instances. Nevertheless, if the number of replicates is small (say lower than 5), Algorithm 1 must be slightly modified, as its minimal distinguishable α value becomes too small. The algorithm for small number of replicates given in [2] shuffles the observations within the instances thus enlarging the number of considered permutations while guaranteeing that each permutation has equal probability to appear. For rank-based tests the Friedman test is extended to this design by an appropriate adjustment of the test statistics [3].

As in Design 1, once the algorithms are found to behave differently one may proceed to the all-pairwise comparisons. In the parametric case, the Tukey and Scheffe's procedures are easily extended. With permutation methods the MSD intervals are computed through Algorithm 2. Differently from Design 1, the synchronized permutations of Algorithm 1 are now necessary. MSD intervals for rank-based tests are again obtained by a variation of the Friedman's test [3].

Note that if there are interactions between instances (for example, if the instances stem from instance classes with different characteristics) it may still be reasonable to infer an average performance over the instances. Nevertheless, this procedure may hide the presence of differences or lead to wrong inference when instances are not balanced through the classes. In this case it is more correct to consider the presence of *stratification variables* such as instance size or instance structure, and report a separate analysis for each combination of these variables.

3 An example analysis on the Graph Coloring Problem

We exemplify the methods presented above on the comparison of algorithms for the well-known Graph Coloring Problem. We compare 9 algorithms, comprising state-of-art and newly developed algorithms, on the class of Flat random graphs [8]. Only six such graphs are available

Procedure Compute_all-pairwise_MSD();
Choose an estimated error ϵ related with B;
Start: Choose a positive number MSD;
for all (i, j) of the $k(k-1)/2$ pairwise comparisons **do**
  if Compute_pairwise_MSD(i, j) returns false **then** goto Start;
end
Return MSD.

Procedure Compute_pairwise_MSD(i, j);

1. Subtract $\bar{X}_{hi.} - \bar{X}_{hj.} + \text{MSD}$ from every value of the data group relative to one of the two algorithms, say i, obtaining the new vector $X_{hit}(\text{MSD}) = X_{hit} - \bar{X}_{hi.} + \bar{X}_{hj.} - \text{MSD}$, $i = 1, 2, ..., n$. This vector is combined with the vector X_{hjt} and constitutes the pool of observations to permute.
2. Compute the statistic $S(\text{MSD})$ for the observed response: $S_0(\text{MSD}) = \sum_h \bar{X}_{hi.}(\text{MSD}) - \bar{X}_{hj.}(\text{MSD})$.
3. By rearranging the observations B times obtain the permutation distribution of the statistic $S^*(\text{MSD}) : \sum_h (\bar{X}^*_{hi.}(\text{MSD}) - \bar{X}^*_{hj.}(\text{MSD}))$.
4. Return "true" if the condition $|\#\{S^*(\text{MSD}) \leq S_0(\text{MSD})\}/B - \alpha_{CW}/2| < \epsilon/2$ is satisfied, else return "false".

Algorithm 2: An algorithm for computing simultaneous MSDs in the case of repeated measures. The algorithm allows to set the comparison-wise α_{CW}-level to α.

online[1] as benchmarking instances, therefore, we select the experimental Design 2 with 10 trials per instance. Figure 3 reports the simultaneous confidence intervals from parametric, permutation and rank-based tests; the assumption of normality for the application of the parametric approach is actually violated; permutation intervals are obtained with B equal to 2000. In Figure 3, the difference between any two algorithms is statistically significant if their confidence intervals do not overlap.

The following are the main observations from the analysis. (i) At the same α-level, permutation tests are slightly more powerful than rank-based tests while ANOVA is more powerful than both but the α level could not be guaranteed because its assumption is violated. (ii) The graphical representation of Figure 3 allows an easy and immediate visual inspection of differences among algorithms and is an helpful tool to convey data and support tables with numerical results, like Table 1. (iii) Rank-based tests may indicate different results due to the particular transformation of data and the ultimate decision about which test is appropriate depends on the final application.

The analysis presented in Figure 3 remains, however, based on the assumption that the results of the algorithms are homoschedastic. In the case in which this assumption is also violated, a very likely case when comparing algorithms with different characteristics, the Algorithm 2 for computing MSDs becomes too conservative. In this case also ANOVA and rank-based tests are inappropriate and alternative methods, that take this effect into account, need to be considered.

[1] M. Trick. "Computational Series: Graph Coloring and its Generalizations", `http://mat.gsia.cmu.edu/COLOR04/`. Last visited March 2005.

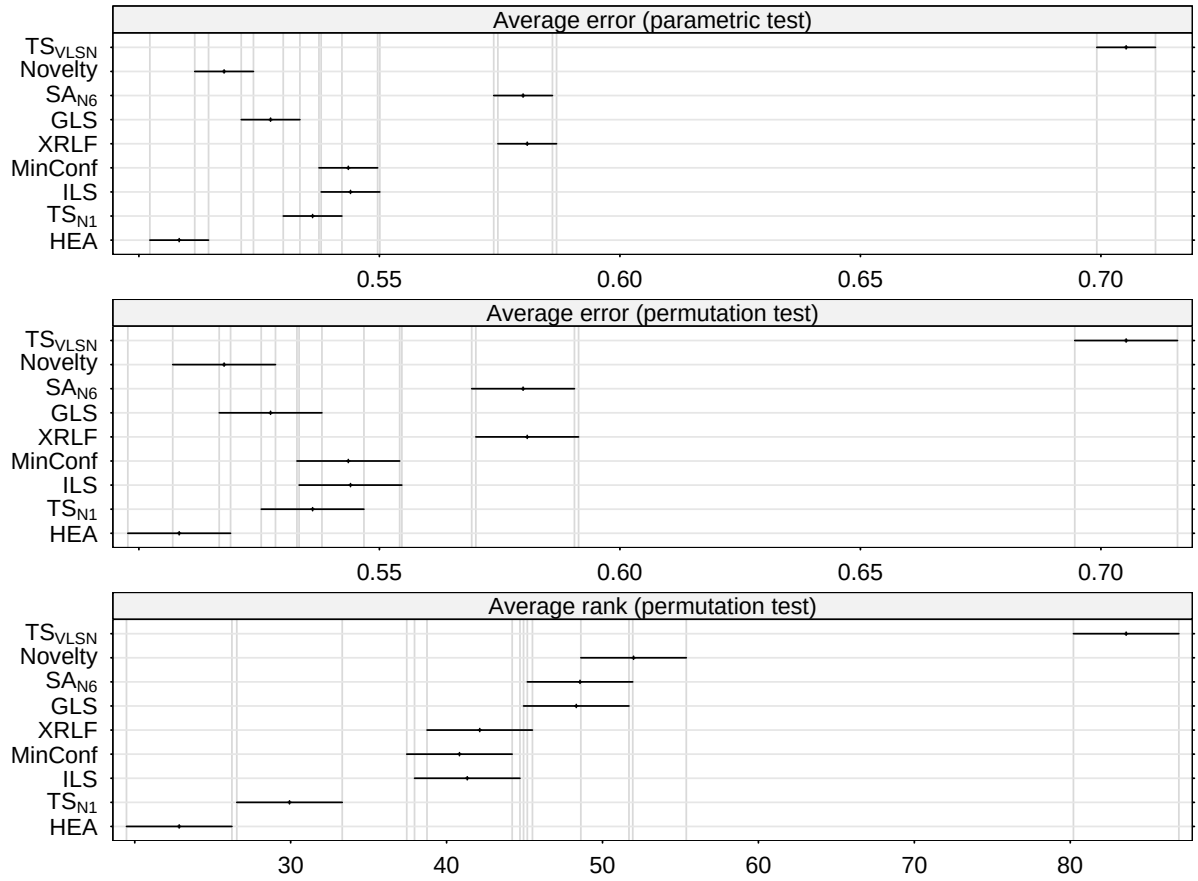

Figure 3: Confidence intervals for all-pairwise comparisons of stochastic algorithms for the Graph Coloring Problem. Three statistical approaches are proposed: parametric, permutation and rank-based. The first two procedures are based on the average error measure e_2, while the latter is based on average ranks. These measures are reported on the x-axis.

Table 1: Numerical results in terms of approximate chromatic number. For each sample of 10 runs per algorithm we report the best result, the first quartile, the median, the third quartile and the worst result.

Instance	HEA	TS_{N1}	ILS	MinConf	XRLF
flat300_20_0	20/20/20/20/20	20/20/20/20/20	20/20/20/20/20	20/20/20/20/20	20/20/20/21/22
flat300_26_0	26/26/26/26/26	26/26/26/26/26	26/26/26/26/26	26/26/26/26/26	33/34/34/34/35
flat300_28_0	31/31/31/32/32	31/31/32/32/32	31/32/32/32/32	31/32/32/32/32	33/34/34/34/34
flat1000_50_0	50/50/78/81/82	85/85/86/87/88	88/88/88/89/90	87/87/88/89/90	84/86/86/87/87
flat1000_60_0	87/87/88/88/89	88/88/89/89/89	89/90/90/90/90	89/89/90/90/90	87/87/87/88/88
flat1000_76_0	88/89/89/89/89	88/89/89/89/90	89/90/90/90/90	90/90/90/90/91	87/87/87/88/89
	GLS	SA_{N2}	Novelty	TS_{N3}	
flat300_20_0	20/20/20/20/20	20/20/20/20/20	22/23/23/24/24	33/34/34/35/35	
flat300_26_0	33/33/33/33/33	32/33/33/33/34	29/29/31/33/34	35/35/35/36/36	
flat300_28_0	33/33/33/33/34	33/33/33/33/34	35/35/35/35/35	35/35/36/36/37	
flat1000_50_0	50/50/50/50/50	86/87/88/89/89	54/54/54/55/55	95/96/96/97/97	
flat1000_60_0	90/90/91/92/93	88/89/89/90/91	64/64/65/65/66	96/97/97/97/97	
flat1000_76_0	92/92/92/92/93	89/89/90/90/90	98/98/98/98/99	96/97/97/97/98	

References

[1] Birattari, M. (2004): "On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs?" Technical report TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Bruxelles, Belgium.

[2] Chiarandini, M. (2005) *Stochastic Local Search Methods for Highly Constrained Combinatorial Optimisation Problems*. PhD Thesis. Computer Science Department. Technische Universität Darmstadt, Darmstadt, Germany.

[3] Conover, W.J. (1999): *Practical nonparametric statistics*. Wiley Series in Probability and statistics. John Wiley & Sons, New York. Third edition.

[4] Dean, A. and Voss, D. (1999): *Design and Analysis of experiments*. Springer Verlag, New York, USA.

[5] Good, P. (2000): *Permutations tests: a practical guide to resempling methods for testing hypotheses*. Springer Verlag, New York, USA. Second edition.

[6] Hollander, M. and Wolfe, D.A. (1999): *Nonparametric Statistical Methods*. John Wiley & Sons, New York, USA. Second edition.

[7] Hsu, J.C. (1996): *Multiple Comparisons. Theory and Methods.* Chapman & Hall, London, UK.

[8] Johnson, D.S. and Trick, M.A. (eds.) (1996): *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI, USA.

[9] Montgomery, D.C. (2000): *Design and Analysis of Experiments*. John Wiley & Sons, New York, USA. Fifth edition.

[10] Pesarin, F. (2001): *Multivariate Permutation Tests. With applications in Biostatistics.* John Wiley & Sons, New York, USA.

[11] Sheskin, D.J. (2000): *Handbook of Parametric and Nonparametric statistical procedures.* Chapman & Hall. Second edition.

[12] Zemel, E. (1981): "Measuring the quality of approximate solutions to zero-one programming problems." *Mathematics of Operations Research*, 6(3). 319–332.