Missouri Estimation of Distribution Algorithms Laboratory

Influence of Selection on Structure Learning in Markov Network EDAs: An Empirical Study

Alexander E. I. Brownlee, John A. W. McCall, and Martin Pelikan

MEDAL Report No. 2012006

April 2012

Abstract

Learning a good model structure is important to the efficient solving of problems by estimation of distribution algorithms. In this paper we present the results of a series of experiments, applying a structure learning algorithm for undirected probabilistic graphical models based on statistical dependency tests to three fitness functions with different selection operators, proportions and pressures. The number of spurious interactions found by the algorithm are measured and reported. Truncation selection, and its complement (selecting only low fitness solutions) prove quite robust, resulting in a similar number of spurious dependencies regardless of selection pressure. In contrast, tournament and fitness proportionate selection are strongly affected by the selection proportion and pressure.

Keywords

Selection, structure learning, spurious dependencies, Markov network, EDA.

Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis
One University Blvd., St. Louis, MO 63121
E-mail: medal@medal-lab.org
WWW: http://medal-lab.org/

Influence of Selection on Structure Learning in Markov Network EDAs: An Empirical Study

Alexander E.I. Brownlee
School of Civil and Building Engineering
Loughborough University
Leicestershire, UK
a.e.i.brownlee@lboro.ac.uk

John A. W. McCall
IDEAS Research Institute
Robert Gordon University
Aberdeen, UK
j.mccall@rgu.ac.uk

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Mathematics and Computer Science
Univ. of Missouri in St. Louis
St. Louis, MO, USA
martin@martinpelikan.net

ABSTRACT

Learning a good model structure is important to the efficient solving of problems by estimation of distribution algorithms. In this paper we present the results of a series of experiments, applying a structure learning algorithm for undirected probabilistic graphical models based on statistical dependency tests to three fitness functions with different selection operators, proportions and pressures. The number of spurious interactions found by the algorithm are measured and reported. Truncation selection, and its complement (selecting only low fitness solutions) prove quite robust, resulting in a similar number of spurious dependencies regardless of selection pressure. In contrast, tournament and fitness proportionate selection are strongly affected by the selection proportion and pressure.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Experimentation, Performance

Keywords

Selection, structure learning, spurious dependencies, Markov network, EDA

1. INTRODUCTION AND CONTEXT

Estimation of Distribution Algorithms (EDA) [9, 13, 16] construct and sample a probability distribution to generate new solutions and move towards an optimum for a problem. A major issue in this field of research is the exploration of factors which influence the quality of the model [30, 31]. This includes the interactions between variables in the model (structure) [8, 10, 12, 19, 28, 29, 31]. Many EDAs are able to learn the structure of the probabilistic model, and in the discrete domain there are two potential sources of error in the structures learned: missing important dependencies and including erroneous (spurious) dependencies [19]. In [37] it was shown that spurious dependencies can negatively impact on performance; recent works including [15, 25] have explored the impact of such errors on the scalability and efficiency of EDAs and the latter of these showed that spurious dependencies can hamper performance, particularly when niching is added to the algorithm.

EDAs effectively use observed likelihoods of variable value combinations in a selected sample of relatively high fitness as an estimate of a distribution that generates high fitness solutions with high probability. Dependencies generated by selection may genuinely reflect such a distribution or may simply be due to a sampling bias imposed by the selection (in [24] this was demonstrated for tournament selection). We term the latter "spurious" dependencies. As well as introducing a potential source of error into the probabilistic model, these are a problem because more complex models require more effort to estimate parameter values – both in population size and estimating the parameters from the data in the population.

There has been some interest over the past few years in EDAs using undirected probabilistic graphical models, or Markov networks [3, 27, 32, 33, 35]. This paper investigates the impact of selection on the discovery of spurious dependencies in this context. Specifically, we apply a simple dependency test structure learning algorithm from [5,26,34] to three well-known fitness functions (onemax, checkerboard and trap-5) and measure the number of spurious dependencies found when using one of nine selection operators including truncation, tournament and fitness proportionate selection. Similar to ref. [24], in this study we focus on the first generation, allowing us to gain better understanding of the operators in the easily comprehended conditions of a uniformly distributed population. The focus of this work on undirected networks contrasts with much of the previous work on selection and structure learning, which has focused on directed (Bayesian) networks. The requirements for the structure types are quite different: an undirected network simply needs to coincide with the additively decomposable function structure of the problem (for example, Ising-DEUM [35]), whereas a directed model structure may need interactions covering distances of 2 nodes or more [10]. Further, in contrast to ref. [6], in this study we focus on the effect of selection on structure learning rather than model parameters, and cover a wider range of commonly-used selection operators.

Related work on selection in EDAs includes a number of methods for selecting low-fitness solutions alongside high-

1: Initialise dependency graph (fully connected)
2: Generate a random population P of size N
3: Select proportion p of P to form σ, where $\sigma \subseteq P$
4: **for all** possible pairs (X_i, X_j) of variables in the problem **do**
5: Compute the Chi-square value χ for (X_i, X_j) over σ
6: **if** $\chi^2 < T$ **then**
7: Remove edge connecting (X_i, X_j) from the dependency graph
8: **end if**
9: **end for**

Figure 1: Structure learning algorithm

fitness ones or the whole population [6, 11, 14, 18, 21, 36]. In this study we also include two variants of truncation selection (taken from ref. [6]) designed to explicitly include low-fitness solutions in structure learning.

The rest of the paper is structured as follows. Section 2 introduces the structure learning algorithm at the centre of our experiments. Section 3 summarises the selection operators studied, and Section 4 outlines the experimental procedure. Sections 5 to 7 present the results. Finally, in Section 8 we draw our conclusions.

2. STRUCTURE LEARNING

The structure learning algorithm is given in Figure 1. Selection is performed to obtain a subset of the population; using the selected solutions as a dataset, pairs of variables are subjected to a statistical independence test to determine whether they are independent of each other. There are many methods which may be used to determine such independence; examples of this are the joint entropy of the pair as used in MIMIC [7] and the Chi-square independence test [17], as used in BMDA [23] and MN-FDA [26]. Each starts with a fully connected graphical model and removes edges where the independence score is below some threshold.

The Chi-square (χ^2) test is a comparison between the joint distribution of a pair of variables and the product of their marginal distributions. If these are equal then the χ^2 value is zero and the variables are said to be independent. Normally, rather than comparison to zero, a threshold T is used to reduce false dependencies caused by noise in the sample data. The threshold we use is 3.84, where the variables may be said to be independent with 95% confidence [2]. For each possible interaction, the test between a pair of variables X_i and X_j is calculated over all values x_i and x_j in a population of N points as in (1).

$$\chi^2_{i,j} = \sum_{x_i, x_j} \frac{(Np(x_i, x_j) - Np(x_i)p(x_j))^2}{Np(x_i)p(x_j)} \qquad (1)$$

Chi-square may also be extended to test higher order interactions by testing for conditional probabilities, but that is not covered further here.

Once completed, the dependency graph will include a set of bivariate dependencies which can be used as the structure for the undirected probabilistic model. In [27], the graph is refined (some dependencies removed) and a search for higher-order cliques is run prior to fitting model parameters. In our experiments we omit these stages as we are only interested in the effect of selection on introducing spurious dependencies to the graph.

3. SELECTION OPERATORS

Several variants of well-known selection operators were implemented in this study.

3.1 Truncation selection

Truncation selection is often used in EDAs, and we implemented the three variants of it described in [6]. In the unaltered form of truncation selection [20], the population is sorted in descending order of fitness, and the top $p * N$ solutions are selected, where $0 < p \leq 1$ and N is the population size. Truncation selection selects a solution only once (unless it is duplicated in the population), and poor fitness individuals are never selected. In this unaltered form, truncation selection is referred to here as *top selection*, as it selects the top part of the population.

As a number of studies [6,11,14,18,21,36] have shown that including low-fitness solutions can be beneficial to model building, we have included two variants of truncation selection which explicitly include such solutions. These are *bottom* selection and *top+bottom selection*. Bottom selection is a duplicate of top selection, but the population is sorted in ascending order of fitness, so it selects the least fit members of the population. Top+bottom selection selects the fittest $(p/2) * N$ and least fit $(p/2) * N$. In [6], it was found that in some situations, top+bottom selection helped to sharpen the information held within a population, resulting in a better model.

3.2 Tournament selection

Several variants of tournament selection (with replacement) are implemented, with tournament sizes of 2, 3, 5 and 10. These are referred to as TS_2, TS_3, TS_5 and TS_{10}. The tournament size provides direct control on selection pressure. The tournament was repeated N times to fill the pool of solutions, with the possibility that solutions could be selected more than once.

3.3 Fitness proportionate selection

Two variants of fitness proportionate (or roulette wheel) selection were implemented for the study. For both, the draw was repeated N times to fill the pool of solutions. Straight fitness proportionate selection has the probability p_i of a solution being directly proportional to its fitness f_i, relative to the fitnesses f_1 to f_N of all other solutions in the population. The probability of selection is explicitly defined in (2).

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i} \qquad (2)$$

Rank selection is the same as fitness proportionate selection, but with the raw fitness values replaced by the fitness ranks for all solutions. Selection pressure for fitness proportionate selection is controlled by how the raw fitness values are treated; in this case, whether raw fitnesses or ranks are used.

3.4 No selection

When the number of selected solutions equals the population size, with the truncation selection variants we effectively

1: **for** each selection operator S **do**
2: **for** each selection proportion p **do**
3: Run the structure learning algorithm, using S with proportion p
4: Compare learned structure with known true structure – how many false interactions were found?
5: **end for**
6: **end for**

Figure 2: Experimental procedure

have no selection. This does not apply for the other selection operators: there will be some duplication or bias among the selected population in this case as they do not guarantee against selecting the same solution more than once.

4. EXPERIMENTS

In order to explore the issue of spurious dependencies introduced by selection, we apply the structure learning algorithm using different selection operators to three well-known fitness functions which have known variable (in)dependencies. We compare the learned structure with the "true" structure; variables known to interact from the definition of the problem. The true structures are described with each problem. The experiments are designed to simultaneously measure the effect of selection operator, selection proportion, and selection pressure on the number of spurious dependencies found.

All three benchmark problems have a bit string representation. These are onemax, checkerboard and trap-5, and they are briefly summarised with the results for each. These problems are selected for their simplicity: they are well understood and have clearly defined variable interactions. The experiments covered a range of problem sizes from 10 bits to 1000 bits for onemax and trap-5, and 16 bits to 400 bits for checkerboard. The results discussed are for 100 bit versions of both problems – reflecting the broader trend seen across the wider range of sizes covered by the experiments. Appendix A features plots for the full set of results.

The experiments followed the steps given in Figure 2, and were repeated over 30 runs. We report mean numbers of spurious interactions detected over all these runs. In the figures, error bars showing one standard deviation have been omitted for clarity, because these were small enough in most cases to be obscured by the data points.

In all experiments, the proportions p of the population selected were 0.01, 0.02, 0.05, 0.1, 0.2, 0.28, 0.5, 0.67, 0.83 and 1 (that is, no selection). The starting population size N was always 10000, allowing us to maintain some diversity in the selected set even when selection pressure is high. Results show the different selection operators, for increasing selection proportion (that is, selecting progressively more of the starting population).

5. RESULTS - ONEMAX

This is one of the simplest and most commonly used benchmarks. In this problem the fitness is simply the number of bits with the value 1, expressed formally in (3).

$$f(x) = \sum_{i=1}^{n} x_i \tag{3}$$

Figure 3: "True" structure for a 5-bit instance of onemax; no dependencies between variables

The onemax problem has no interactions between variables (Figure 3), which makes it an excellent candidate for measuring an algorithm's tendency to learn false dependencies. Any dependencies found by the structure learning algorithm for this problem are counted as false dependencies. The raw number of false interactions found for 100 bit onemax for the different selection operators and selection proportions is illustrated in Figure 4.

Several trends can be observed. With a small selection proportion, all the operators produce a similar number of false positives; 200-300 spurious dependencies, and is in line with the demonstration of tournament selection resulting in discovery of dependencies in [24]. Truncation selections seems to perform best; top selection and bottom selection has around the same number of false positives for any selection proportion. This reinforces the findings of works including [6, 11, 14, 18, 21, 36] that low fitness solutions can be useful in constructing the model.

T+B selection appears to be worst among all the operators with a small selection proportion (high pressure). As selection pressure decreases more spurious dependencies are found, peaking around selection of 0.1 of the population before improving as selection pressure drops further. This is in contrast to the results seen in [6], where T+B selection was found to help produce a good model when estimating model parameters. The poorer performance of T+B selection can be explained – in that work, the model incorporated fitness during the parameter estimation phase, so the low and high fitness solutions were still distinguishable. In contrast, here once the selection is performed the difference between high and low fitness solutions is lost – this would imply that the structure learning algorithm would need modified to make proper use of top+bottom selection.

The other operators result in increasing numbers of false interactions as selection proportion increases. This will be for two reasons. When applying these operators with a selection proportion of 1.0 (selecting the whole population) there will be more duplicated individuals in the selected set (the population size being the same as the number to be selected). For TS_2, selecting a set equal to the population size results in 70% of the selected set being solutions with at least one duplicate. For TS_{10} (higher selection pressure than TS_2), this figure is 92%. This creates higher frequencies of specific value pairs which skew the Chi-square values. In contrast, truncation selection of the whole population has no duplication.

More crucially, as the selection proportion increases with tournament selection, the sample size increases but the selection pressure remains the same. As the statistical test is dependent on the sample size, increasing sample size increases the chance of spurious dependencies. When changing the selection proportion with the truncation selection variants, the selection pressure is also changed, which appears to have a cancelling effect.

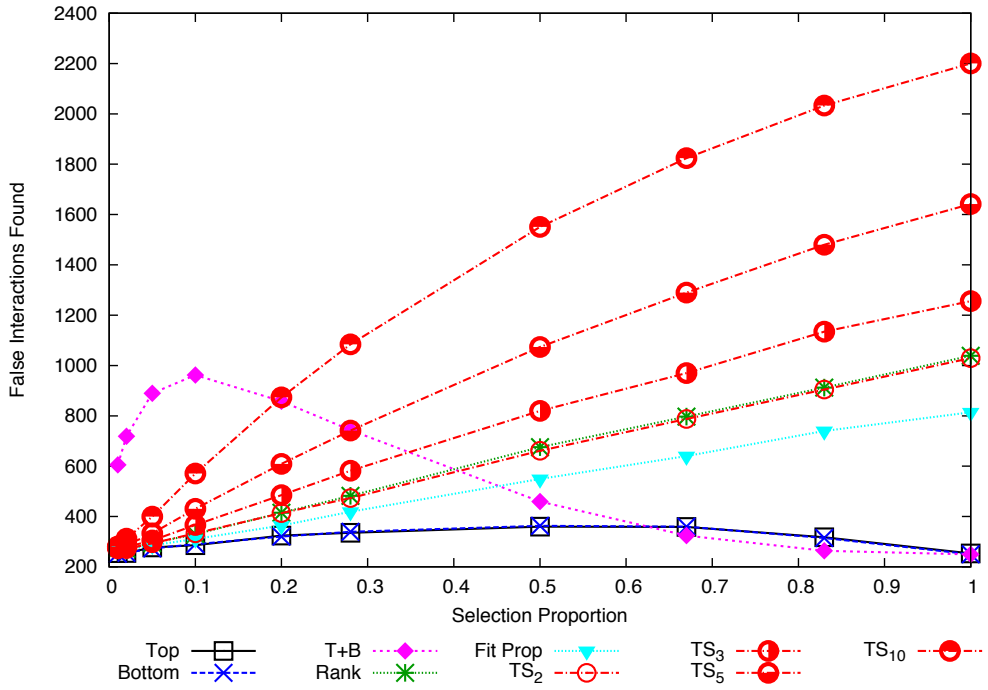

Figure 4: False interactions found for 100 bit onemax with different selection operators and proportions. For clarity, from top to bottom the plots are for TS_{10}, TS_5, TS_3, TS_2 and rank (near identical), fit prop, and finally top selection and bottom selection (near identical). The plot for T+B selection starts high, increases, then falls after a selection proportion of 0.1.

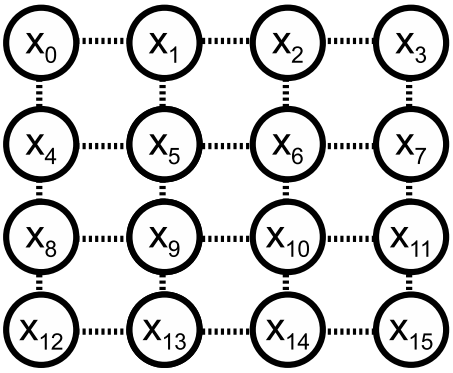

Figure 5: "True" structure for a 16-bit instance of checkerboard; dependencies (dotted lines) between variables form a lattice

6. RESULTS - CHECKERBOARD

The checkerboard problem [1, 13] introduces bivariate interactions. Chromosomes are a square number of bits in length; they represent the rows of a s x s grid concatenated into one string. The objective is to realise a grid with a checkerboard pattern with alternating 1s and 0s; thus each 1 should be surrounded by 0s and vice versa, not including corners. Formally this is written as in (4).

$$f(x) = 4(s-2)^2 - \sum_{i=2}^{s-1}\sum_{j=2}^{s-1}\left\{\begin{array}{r}\delta(x_{ij}, x_{i-1j}) \\ +\,\delta(x_{ij}, x_{i+1j}) \\ +\,\delta(x_{ij}, x_{ij-1}) \\ +\,\delta(x_{ij}, x_{ij+1})\end{array}\right\}, \quad (4)$$

where:

$$\delta_{ij} = \left\{\begin{array}{lll} 1 & if & i = j \\ 0 & if & i \neq j \end{array}\right. \quad (5)$$

This problem is useful as it has a simple underlying 2D lattice structure which is easily understood and visualised. The true structure is perfectly aligned with fitness independence, and for 100 bit checkerboard has 144 interactions: a 2D lattice, with no wrapping around the edges (Figure 5). Any interactions found by the structure learner which were not connecting neighbouring nodes in the lattice were counted as spurious. We can do this for undirected graphical models because such a model simply needs to coincide with the additively decomposable function structure of the problem to be able to generate high fitness solutions with high probability. In contrast, a directed model structure may need interactions covering distances of 2 nodes or more (this at least being the case for Ising spin glasses, which have a similar lattice structure [10]).

The results are illustrated in Figure 6. As with onemax, top selection and bottom selection perform very similarly, resulting in around 200-300 spurious interactions being found regardless of selection proportion.

The other selection operators decrease in performance (more spurious interactions) as the chance of duplicates in the se-

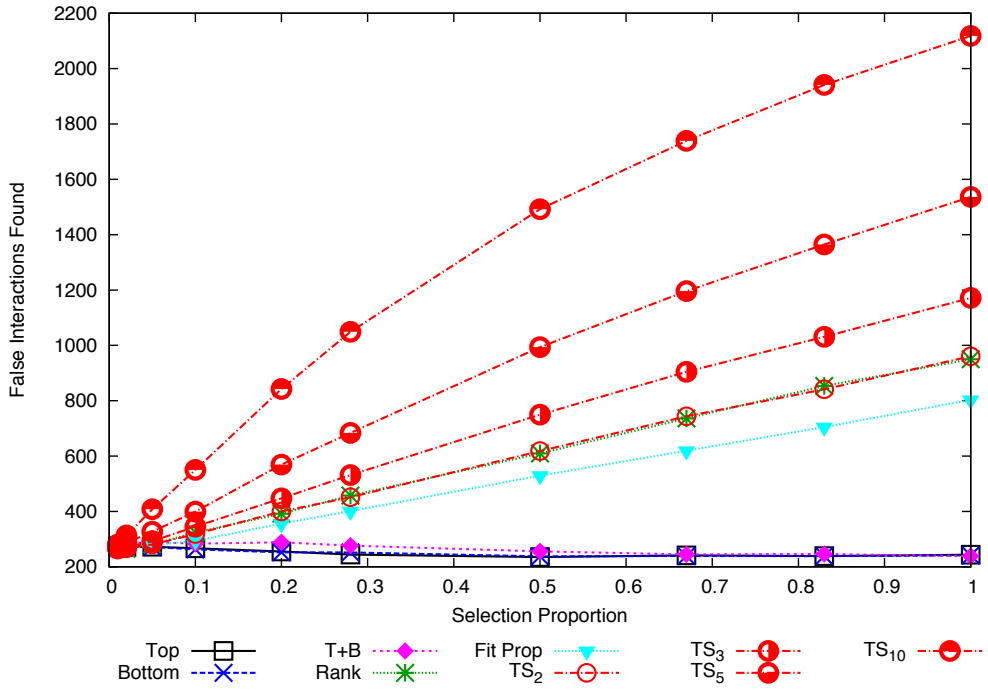

Figure 6: False interactions found for 100 bit checkerboard with different selection operators and proportions. For clarity, from top to bottom the plots are for TS_{10}, TS_5, TS_3, TS_2 and rank (near identical), fit prop, T+B selection, and finally top selection and bottom selection (near identical).

lected set increases – both as the selected proportion increases to select the same number of solutions as the population size, and for tournament selection as the tournament size (selection pressure) increases.

One difference to the trends seen for onemax, is that top+bottom selection does not show the same poor performance when selecting a small proportion of the population – instead resulting in a similar number of spurious dependencies as top selection and bottom selection. A possible explanation for this is that the two optima for checkerboard contain alternating patterns of 0 and 1. Thus, in the selected set there will be lots of alternating patterns at the top (but in different positions), and lots of patterns with adjacent pairs the same at the bottom. This will mean Chi-square often sees little difference from the product of the marginals and thus detects few spurious dependencies.

7. RESULTS - TRAP-5

The trap functions [22] are designed to deceive evolutionary algorithms into converging on a local optimum. This is particularly a problem for algorithms which do not consider interactions between variables. The trap function of order k is defined in (6). It computes fitness by dividing the bit string into blocks.

$$f(x) = \sum_{i=1}^{n/k} trap_k(x_{b_i,1} + ... + x_{b_i,k}) \quad (6)$$

Each block $(x_{b_i,1}, +... + x_{b_i,k})$ gives a fitness, calculated as in (7), where u is the number of 1s in the block of k bits. Trap-5, which is the specific instance used in this paper, has

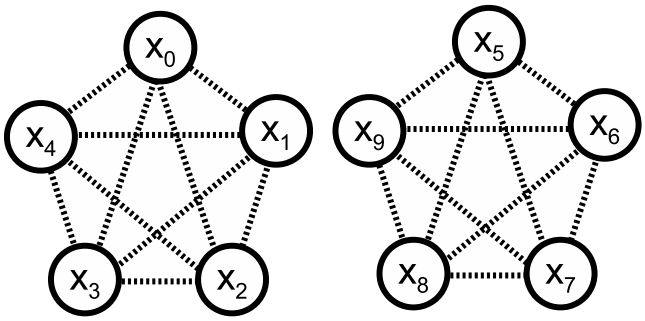

Figure 7: "True" structure for a 10-bit instance of trap-5; dependencies (dotted lines) between variables in each group of five form a fully connected graph, with no dependencies between groups

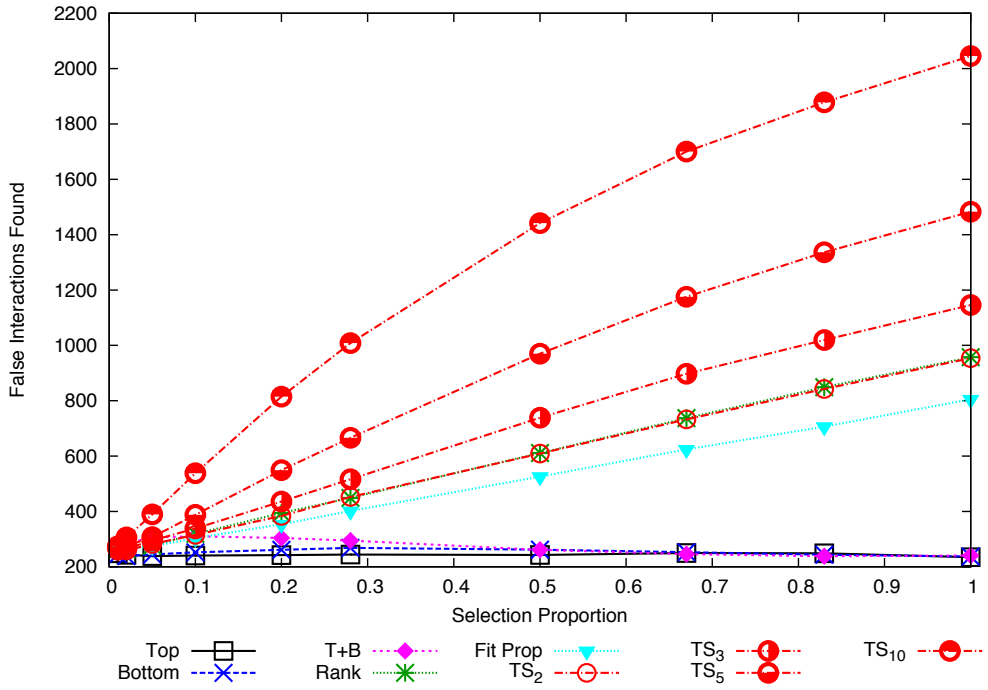

Figure 8: False interactions found for 100 bit trap-5 with different selection operators and proportions. For clarity, from top to bottom the plots are for TS_{10}, TS_5, TS_3, TS_2 and rank (near identical), fit prop, T+B selection, top selection and finally bottom selection.

$k = 5$, $f_{high} = 5$ and $f_{low} = 4$.

$$trap_k(u) = \begin{cases} f_{high} & \text{if} \quad u = k \\ f_{low} - u\frac{f_{low}}{k-1} & otherwise \end{cases} \tag{7}$$

The functions are designed to mislead or *trap* algorithms which do not account for interactions between variables, so is also useful for testing a structure learning algorithm. As u increases, fitness decreases, which leads the algorithm away from the global optimum. Algorithms which do find interactions should be able to determine that groups of k variables being switched to all 0 improves fitness.

For trap-5, we considered the true interactions to be a fully connected graph within each groups of five bits (Figure 7). This means that for each group there are 10 interactions, and 1000 interactions for a 100 bit instance of the problem. We count as spurious interactions falling outside these groups.

The results are similar to those for onemax and checkerboard, and are illustrated in Figure 8. Top selection and bottom selection perform similarly, with 200-300 spurious interactions being found regardless of selection proportion.

As before, the other selection operators result in more spurious interactions being found as the sample size and along with it the chance of duplicates in the selected set increase – as the proportion of the population selected increases. Further, as the tournament size (selection pressure) increases, so does the number of spurious dependencies.

Similar to checkerboard but not onemax, top+bottom selection shows a similar number of spurious dependencies as top selection and bottom selection. Here this is possibly caused by the deceptiveness of the problem – solutions quite close to the optimum (at least within the groups) will have a low fitness, so will appear in the bottom part of the population. This will mean that top+bottom selection effectively becomes top selection for this problem.

8. CONCLUSIONS

We have presented an empirical study of the effect of different selection operators, proportions and pressures on an algorithm for learning the structure of an undirected probabilistic model. Several conclusions can be drawn from these results.

Truncation selection for structure learning return similar number of false interactions for wide range of selection proportions. In addition, the number found is similar for each of the problems. This may seem curious given that all the problems examined have different numbers of true interactions. However, the true interactions make up a small fraction of the total number of possible variable pairs and assuming an even distribution of false interactions among the set of those possible, it is reasonable that there would be a similar number found for each problem. This consistency allows fine tuning of the operator to find as high a number of true interactions as possible, without the issue of introducing more false ones into the model structure. We conclude that truncation selection is quite a robust operator (at least in the context of these problems), further justifying its common use among EDAs. This is a similar conclusion to that in ref. [15], where Lima et al found that truncation selection performs well because the distribution of solutions it selects is uniform, which is more suitable for metrics like K2.

Top selection and bottom selection perform similarly for all the test problems, reinforcing similar results [6, 11, 14, 18,

21,36] that show the usefulness of including low fitness solutions in model construction. Fitness proportionate and rank selection are quite poor for structure learning, particularly as selection proportion is increased. Tournament selection shows a similar pattern: increasing the tournament size (a higher selection pressure) leads to the number of spurious interactions found increasing. This is undesirable as having a small selection proportion necessitates a large population, which will require many fitness evaluations. A likely explanation for this is given in [24], that most selection methods introduce nonlinearities between bits regardless of whether or not these bits are correlated in the fitness function (an exception being Boltzmann selection, which preserves independence between the bits that are not related in the fitness). This problem grows as selection pressure increases. It may also be due to a combination of the increased sample size and a large number of duplicates in the selected set.

Although in this study we used a large population size to allow us to keep some diversity in the selected set with high selection pressures, producing more consistent results, we believe that the insights carry over to smaller population sizes. If the sample size is reduced, the Chi-square values also reduce and so the threshold for the structure learning algorithm must be lowered in order to find the true interactions. In [29], Santana used a population size of 1000 and threshold of 0.75. A repeat of our trap-5 experiment with this population size and threshold showed similar results. Further exploration of this and the wider issues of the interplay between the population size, selection operator or the selection pressure and minimum population size for finding an accurate enough model present an interesting direction for further work. This relates to other similar lines of current research in EDAs such as [25,38].

In this paper, we have focused on the first generation only (similar to ref. [24]). There are many other considerations when choosing a selection operator in an EDA beyond the quality of the model in the first generation. The initial study here does provide some insight into the selection operators studied, and in the easily understood conditions of a uniformly distributed starting population. With these foundations, the logical next step is to study selection in the same way over the course of an evolutionary run, and in parallel explore how selection affects the overall algorithm performance in terms of function evaluations required to find an optimum.

We intend to explore these issues on a much wider range of fitness functions with different characteristics, with the goal of determining approaches to choosing the appropriate selection for a given problem. The series of experiments in this paper provides the basis for a more extensive study which will help to inform the current research on improving model quality and efficiency in EDAs.

9. ACKNOWLEDGEMENT

Martin Pelikan was supported by the National Science Foundation under grants ECS-0547013 and IIS-1115352. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

10. REFERENCES

[1] S. Baluja and S. Davies. Using optimal dependency-trees for combinational optimization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 30–38. Morgan Kaufmann Publishers Inc, 1997.

[2] S. Boslaugh and P. A. Watters. *Statistics: A Desktop Quick Reference*. O'Reilly, 2008.

[3] A. E. I. Brownlee. *Multivariate Markov Networks for Fitness Modelling in an Estimation of Distribution Algorithm*. PhD thesis, Robert Gordon University, Aberdeen, May 2009.

[4] A. E. I. Brownlee, J. A. W. McCall, and M. Pelikan. Influence of selection on structure learning in Markov network EDAs: An empirical study. Technical Report MEDAL Report No. 2012006, Missouri Estimation of Distribution Algorithms Laboratory, Univ. of Missouri in St. Louis, 2012. Online: http://medal-lab.org/files/2012006.pdf.

[5] A. E. I. Brownlee, J. A. W. McCall, S. K. Shakya, and Q. Zhang. Structure Learning and Optimisation in a Markov-network based Estimation of Distribution Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 447–454, Trondheim, Norway, 2009. IEEE Press.

[6] A. E. I. Brownlee, J. A. W. McCall, Q. Zhang, and D. Brown. Approaches to Selection and their effect on Fitness Modeling in an Estimation of Distribution Algorithm. In *Proc. of the IEEE World Congress on Computational Intelligence (CEC 2008)*, pages 2621 –2628, Hong Kong, China, 2008. IEEE Press.

[7] J. S. de Bonet, C. L. Isbell Jr., and P. Viola. MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, 1997.

[8] C. Echegoyen, J. Lozano, R. Santana, and P. Larranaga. Exact Bayesian network learning in estimation of distribution algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 1051–1058, sept. 2007.

[9] M. Hauschild and M. Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111 – 128, 2011.

[10] M. Hauschild, M. Pelikan, K. Sastry, and C. Lima. Analyzing probabilistic models in hierarchical BOA. *IEEE Transactions on Evolutionary Computation*, 13:1199–1217, December 2009.

[11] Y. Hong, G. Zhu, S. Kwong, and Q. Ren. Estimation of distribution algorithms making use of both high quality and low quality individuals. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2009)*, pages 1806–1813, August 2009.

[12] L. Kallel, B. Naudts, and R. Reeves. Properties of fitness functions and search landscapes. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 177–208. Springer Verlag, 2000.

[13] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston, 2002.

[14] X. Li, S. Mabu, and K. Hirasawa. Use of infeasible individuals in probabilistic model building genetic network programming. In *Proc. of the 13th Genetic*

and Evolutionary Computation COnference (GECCO 2011), pages 601–608, New York, NY, USA, 2011. ACM.

[15] C. F. Lima, F. G. Lobo, M. Pelikan, and D. E. Goldberg. Model accuracy in the Bayesian optimization algorithm. *Soft Computing*, 15(7):1351–1371, 2010.

[16] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer-Verlag, 2006.

[17] L. A. Marascuilo and M. McSweeney. *Nonparametric and Distribution-Free Methods for Social Sciences*. Brooks / Cole Publishing Company, California, 1977.

[18] T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary computation based on Bayesian classifiers. *Int'l Jnl. of Applied Mathematics and Computer Science*, 14(3):101–115, 2004.

[19] H. Mühlenbein and T. Mahnig. Evolutionary optimization using graphical models. *New Gen. Comput.*, 18(2):157–166, 2000.

[20] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm I. continuous parameter optimization. *Evolutionary Computuation*, 1(1):25–49, 1993.

[21] M. Munetomo, N. Murao, and K. Akama. Introducing assignment functions to Bayesian optimization algorithms. *Info. Sciences*, 178(1):152 – 163, 2008.

[22] M. Pelikan. *Bayesian optimization algorithm: from single level to hierarchy*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

[23] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, 1999.

[24] M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the Bayesian optimization algorithm. *Int'l. Jnl. of Approximate Reasoning*, 31(3):221 – 258, 2002.

[25] E. Radetic and M. Pelikan. Spurious dependencies and EDA scalability. In *Genetic and Evolutionary Computation Conference*, pages 303–310, 2010.

[26] R. Santana. A Markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning; Lecture Notes in Artificial Intelligence*, volume 2837, pages 337–348, Berlin, 2003. Springer-Verlag.

[27] R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.

[28] R. Santana, C. Bielza, J. A. Lozano, and P. Larrañaga. Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO 2009)*, pages 445–452, New York, NY, USA, 2009. ACM.

[29] R. Santana, P. Larrañaga, and J. A. Lozano. Interactions and dependencies in estimation of distribution algorithms. In *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2005)*, volume 1, pages 1418–1425. IEEE Press, 2-4 Sept. 2005.

[30] R. Santana, P. Larrañaga, and J. A. Lozano. Challenges and open problems in discrete EDAs. Technical Report EHU-KZAA-IK-1/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country, October 2007.

[31] R. Santana, P. Larrañaga, and J. A. Lozano. Research topics on discrete estimation of distribution algorithms. *Memetic Computing*, 1(1):35–54, 2009.

[32] S. Shakya and R. Santana. An EDA based on local Markov property and Gibbs sampling. In *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2008)*, pages 475–476. ACM Press, 2008.

[33] S. Shakya and R. Santana, editors. *Markov Networks in Evolutionary Computation*. Springer, 2012.

[34] S. K. Shakya, A. E. I. Brownlee, J. A. W. McCall, F. Fournier, and G. Owusu. A fully multivariate DEUM algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 479–486. IEEE Press, 2009.

[35] S. K. Shakya and J. A. W. McCall. Optimization by estimation of distribution with DEUM framework based on Markov random fields. *International Journal of Automation and Computing*, 4(3):262–272, 2007.

[36] D. Wallin and C. Ryan. Using over-sampling in a Bayesian classifier EDA to solve deceptive and hierarchical problems. In *Proc. of the IEEE Congress on Evolutionary Computation*, pages 1660 –1667, 2009.

[37] T.-L. Yu, K. Sastry, and D. E. Goldberg. Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1217–1224, New York, NY, USA, 2005. ACM.

[38] T.-L. Yu, K. Sastry, D. E. Goldberg, and M. Pelikan. Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 601–608, New York, NY, USA, 2007. ACM.

APPENDIX

A. FULL RESULTS

This section presents the figures for the experiments with all instances of onemax, checkerboard and trap-5. For the larger instances than those discussed earlier (that is, more than 100 bits), the results are similar to those in the main body of the paper, except that the absolute numbers of spurious dependencies are larger. For the smaller instances, we see more noise in the results (higher standard deviations across the repeat runs). For onemax instances smaller than 100 bits, with the truncation selection variants there is also an increase in spurious dependencies when selecting around half of the population.

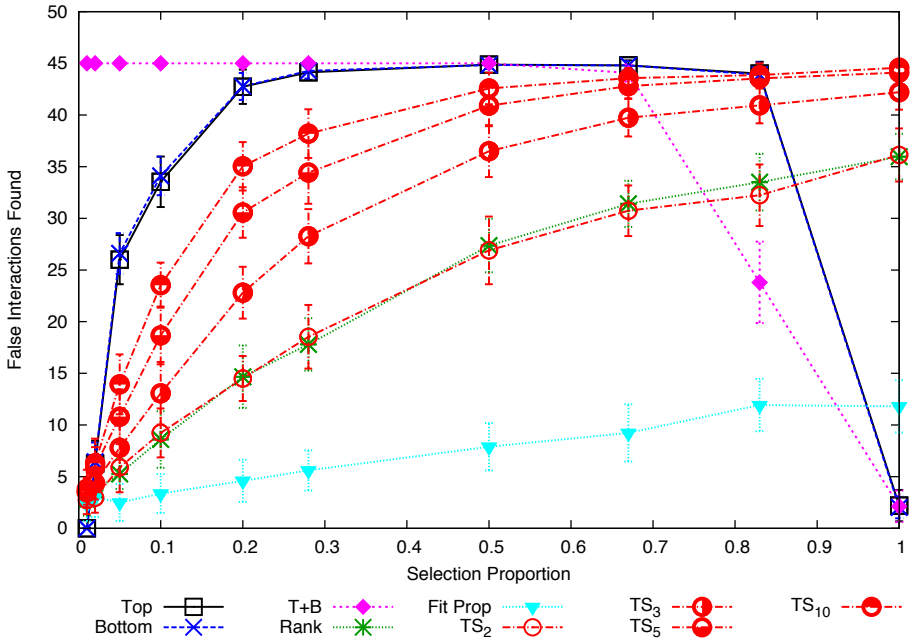

Figure 9: False interactions found for 10 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

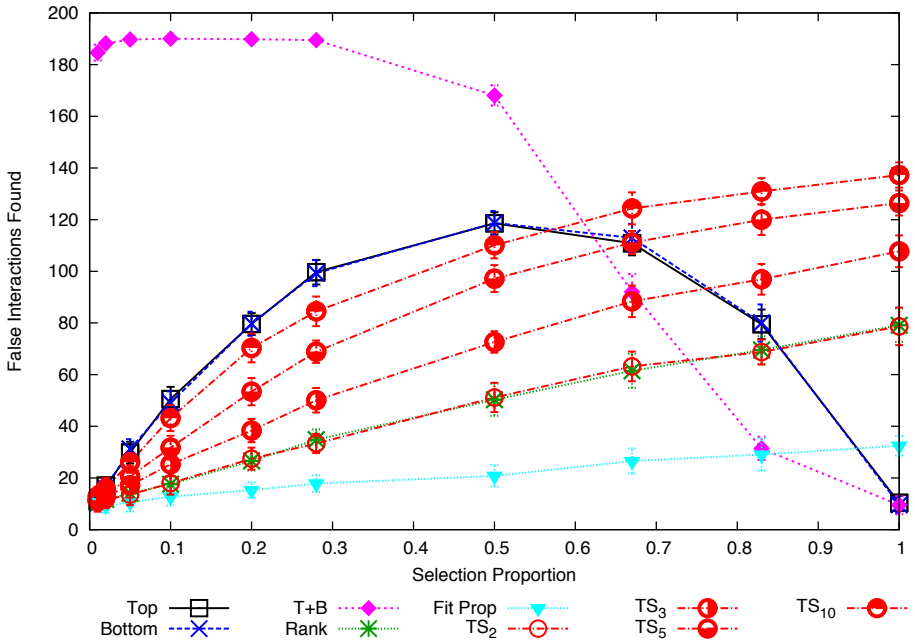

Figure 10: False interactions found for 20 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

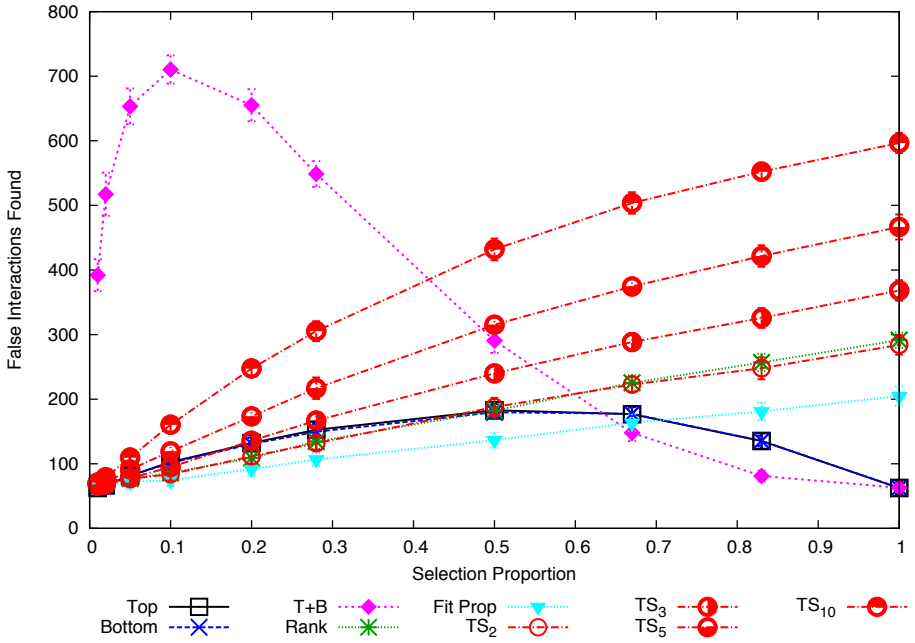

Figure 11: False interactions found for 50 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

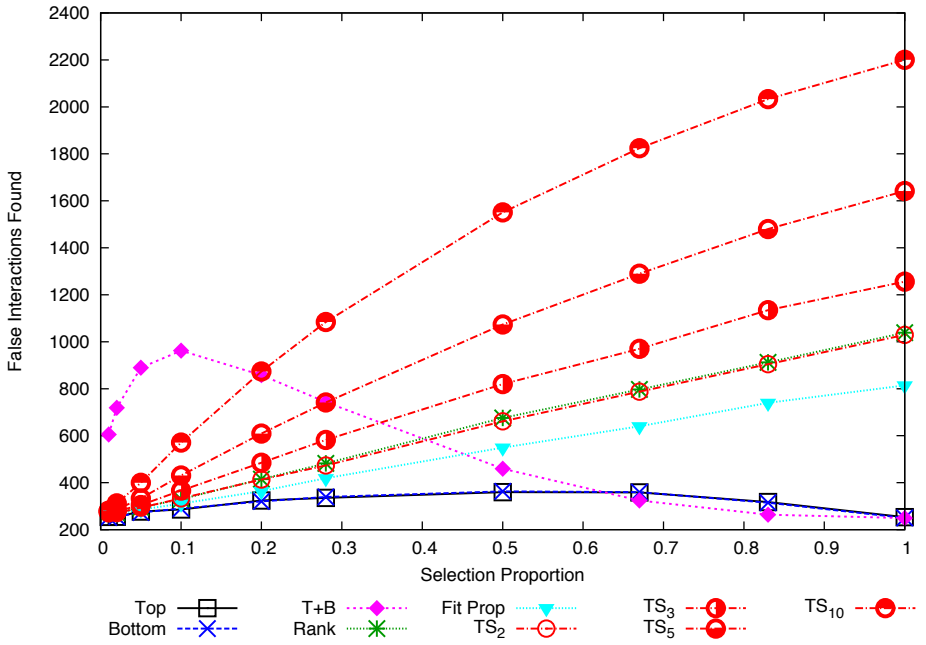

Figure 12: False interactions found for 100 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

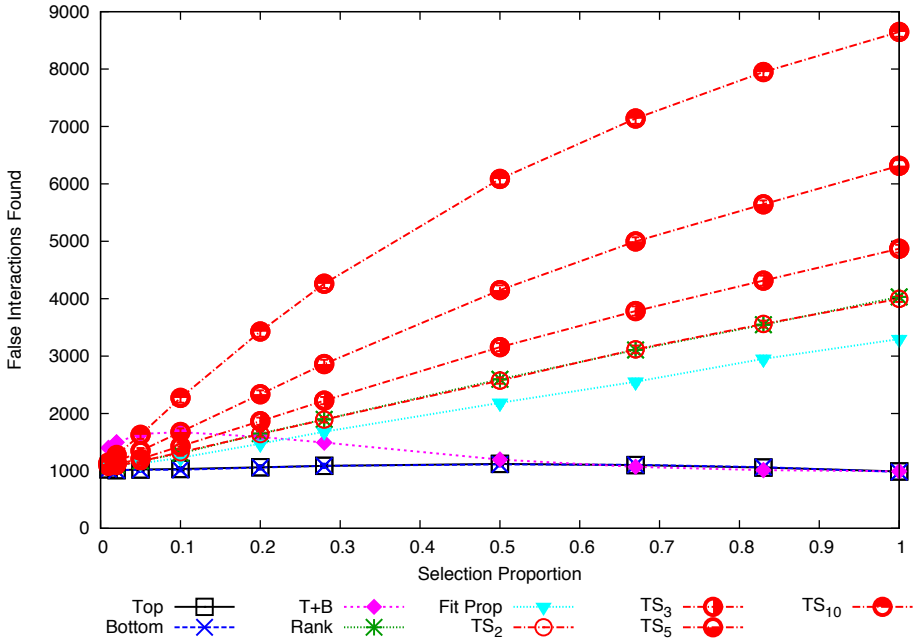

Figure 13: False interactions found for 200 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

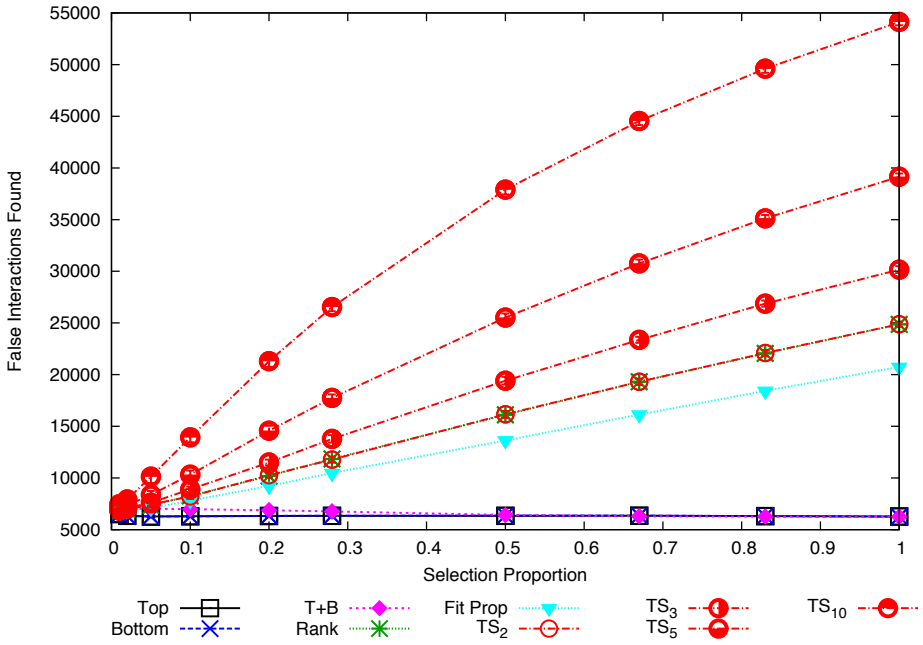

Figure 14: False interactions found for 500 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

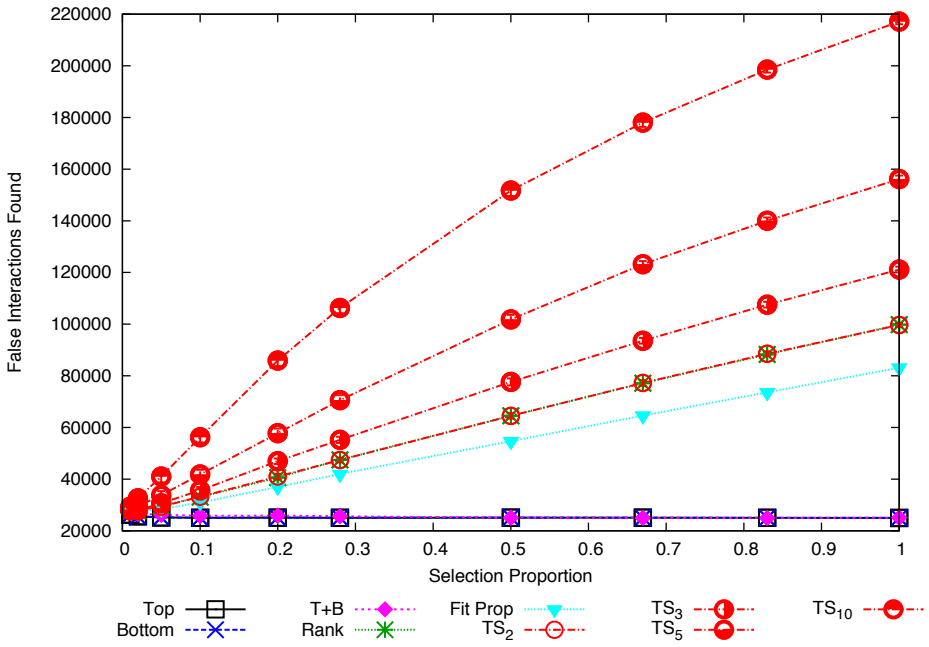

Figure 15: False interactions found for 1000 bit onemax with different selection operators and proportions. Error bars show one standard deviation.

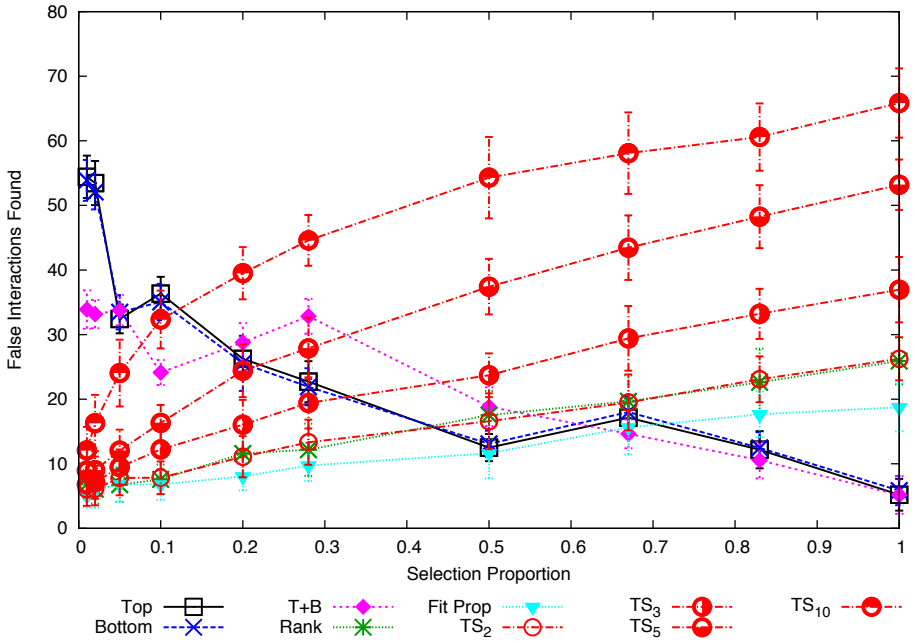

Figure 16: False interactions found for 16 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

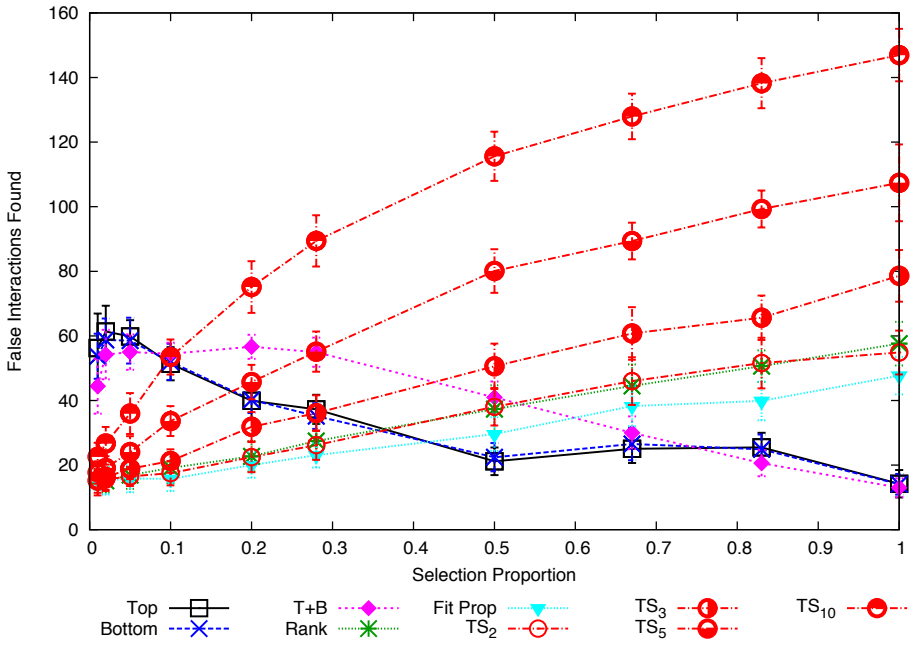

Figure 17: False interactions found for 25 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

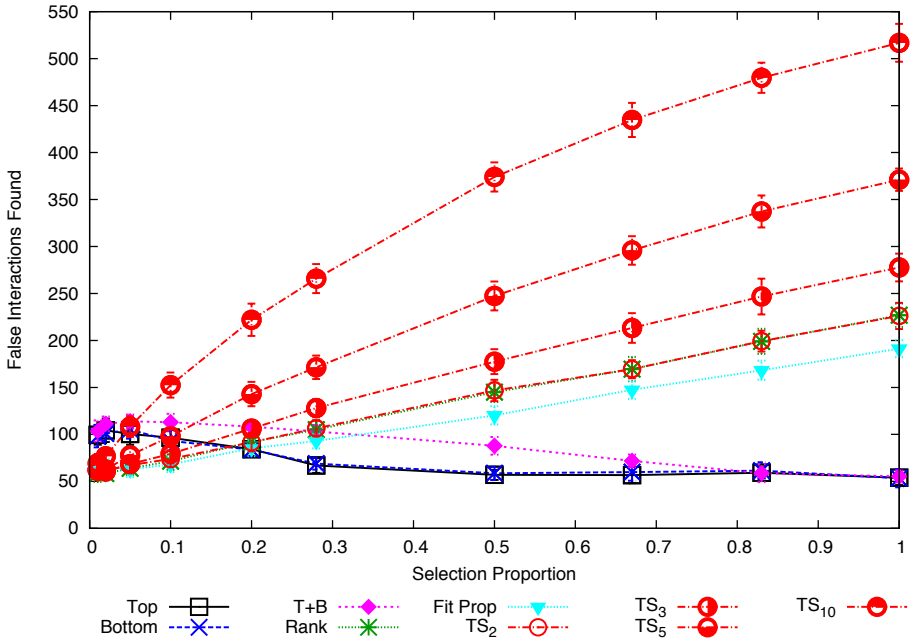

Figure 18: False interactions found for 49 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

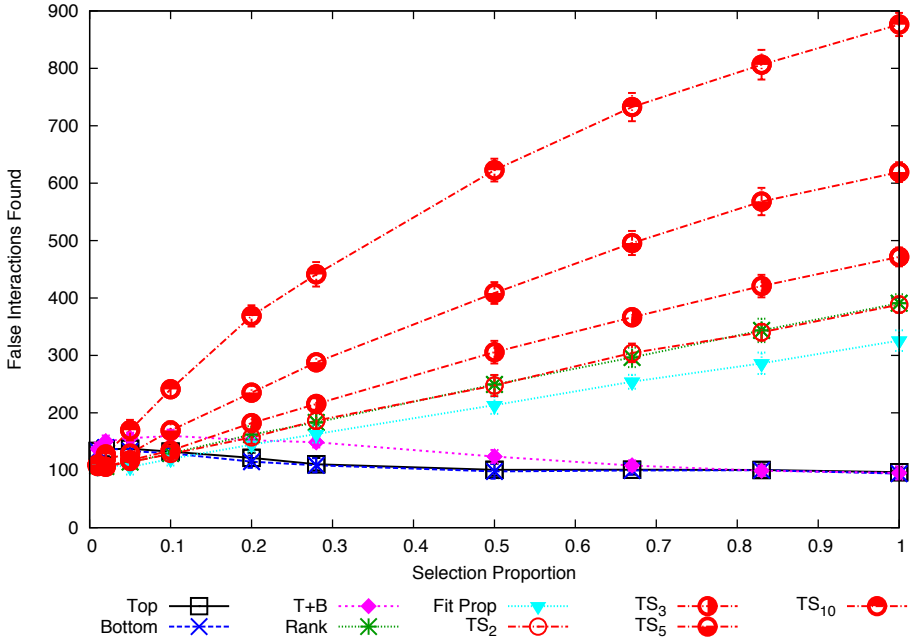

Figure 19: False interactions found for 64 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

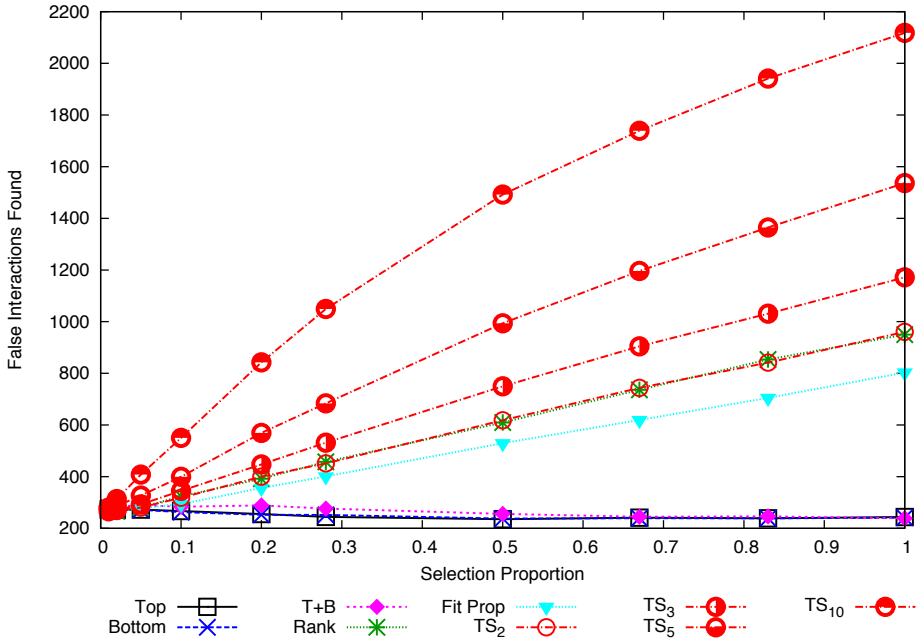

Figure 20: False interactions found for 100 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

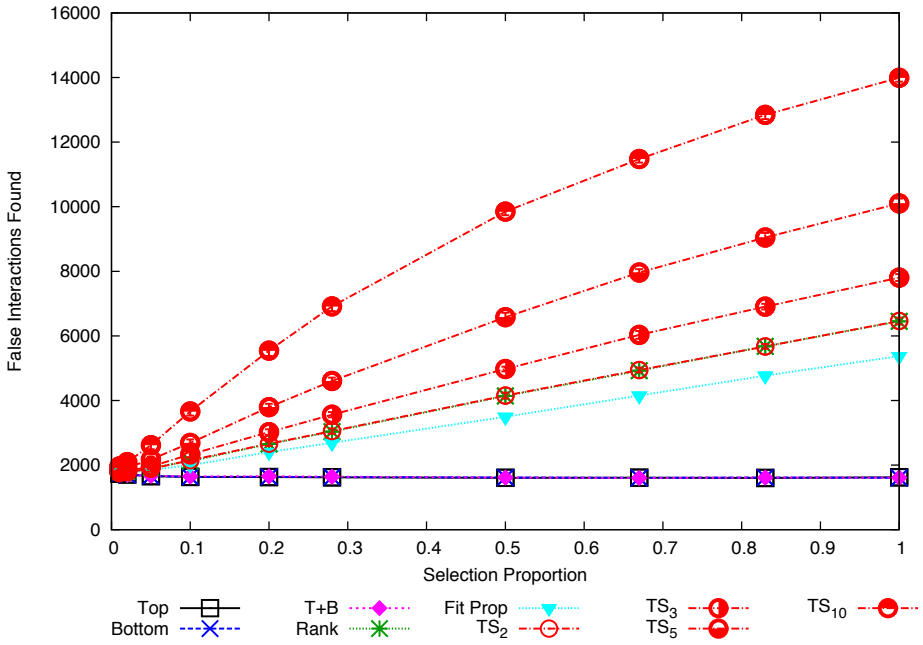

Figure 21: False interactions found for 256 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

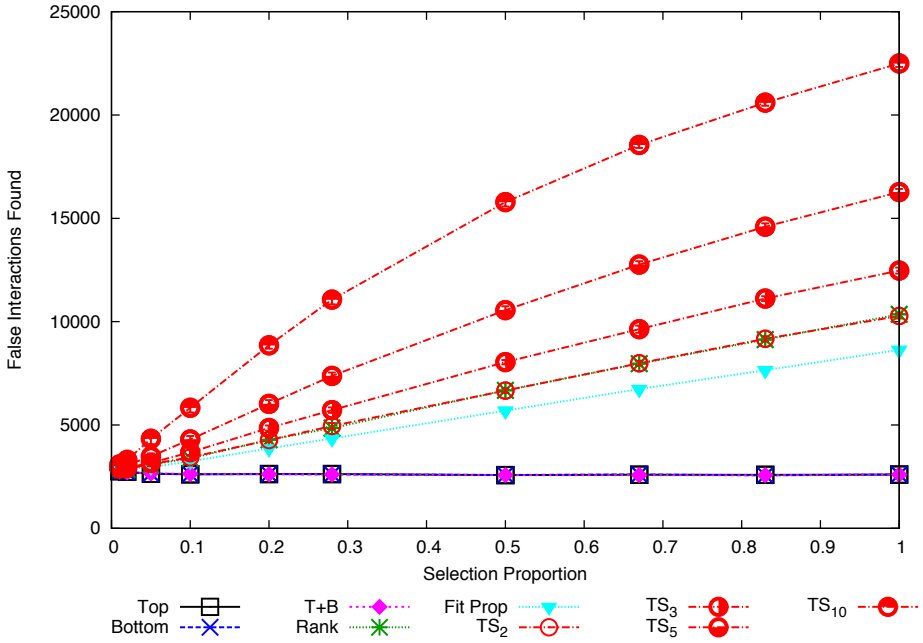

Figure 22: False interactions found for 324 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

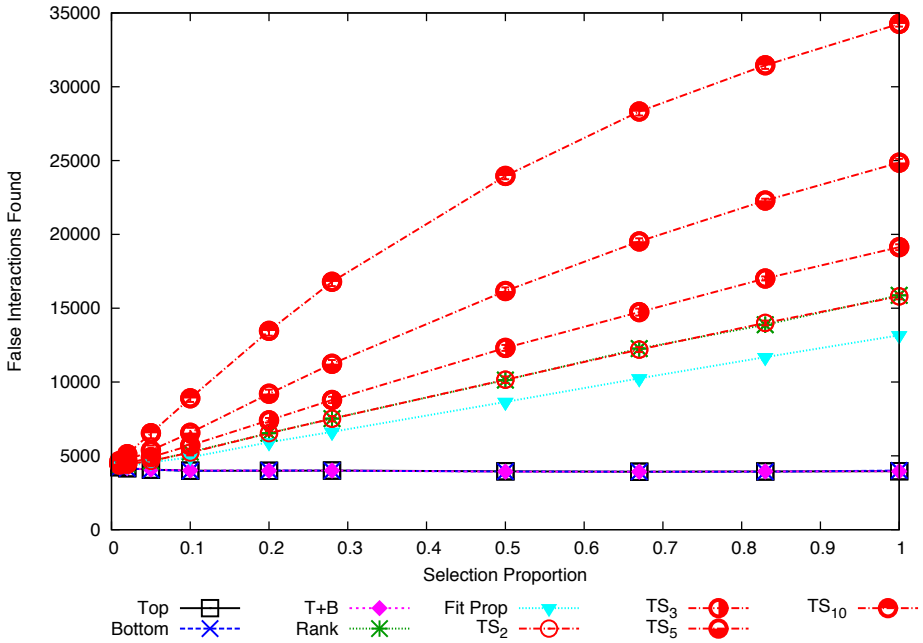

Figure 23: False interactions found for 400 bit checkerboard with different selection operators and proportions. Error bars show one standard deviation.

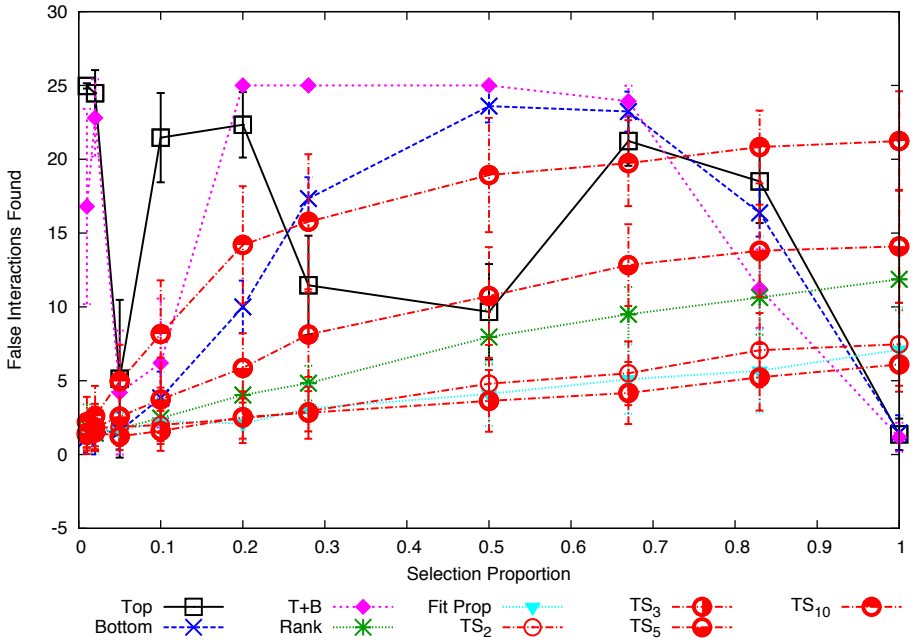

Figure 24: False interactions found for 10 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.

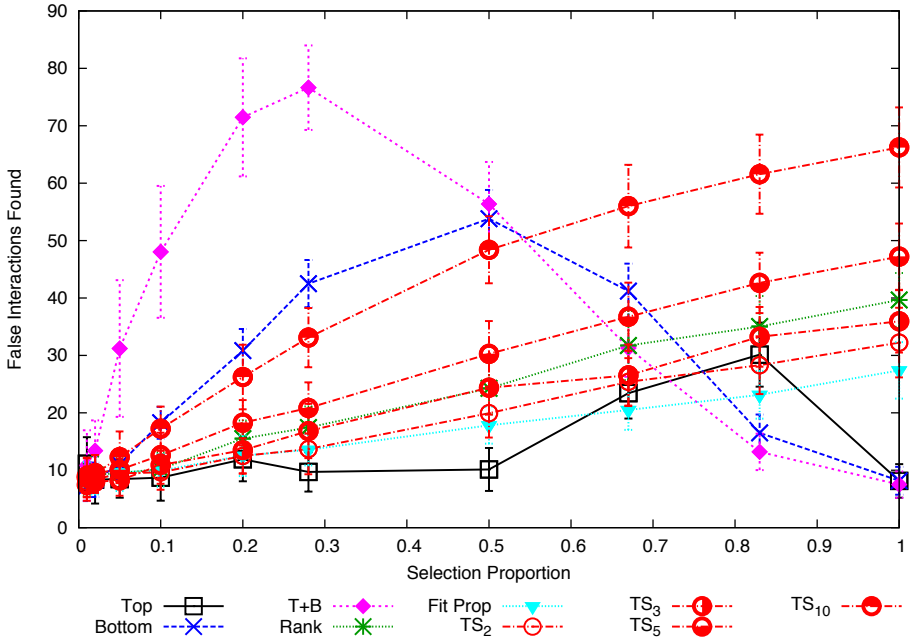

Figure 25: False interactions found for 20 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.

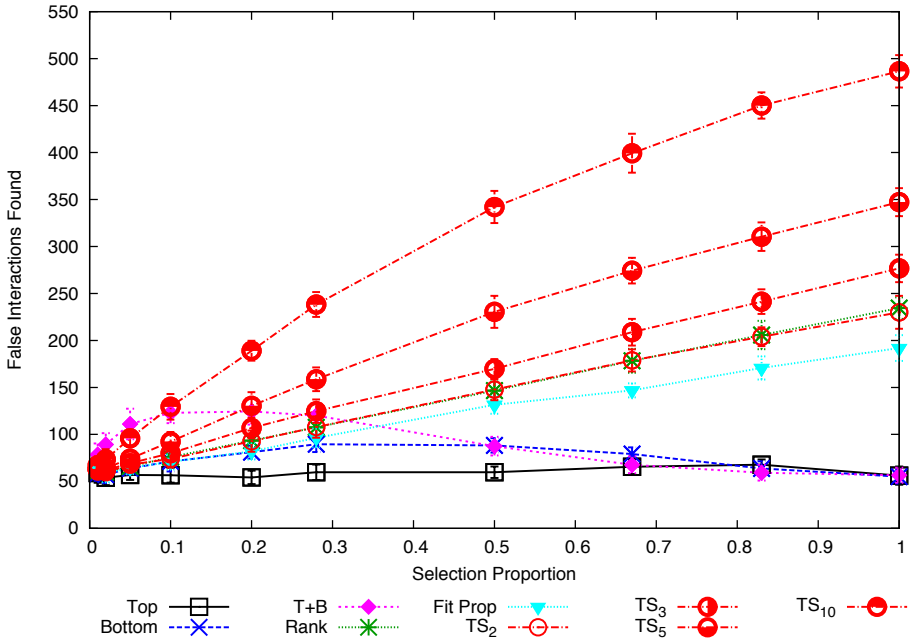

Figure 26: False interactions found for 50 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.

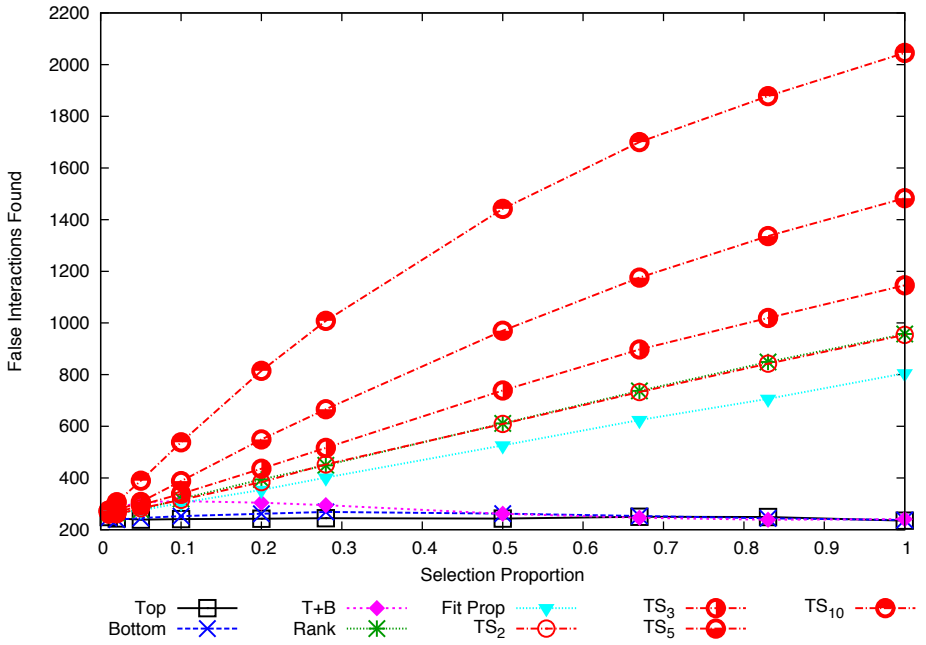

Figure 27: False interactions found for 100 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.

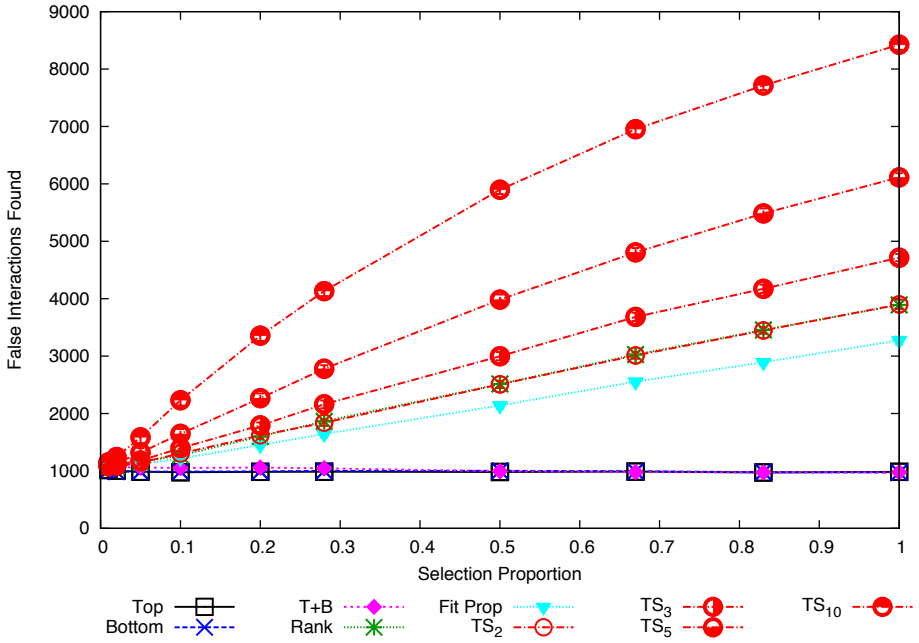

Figure 28: False interactions found for 200 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.

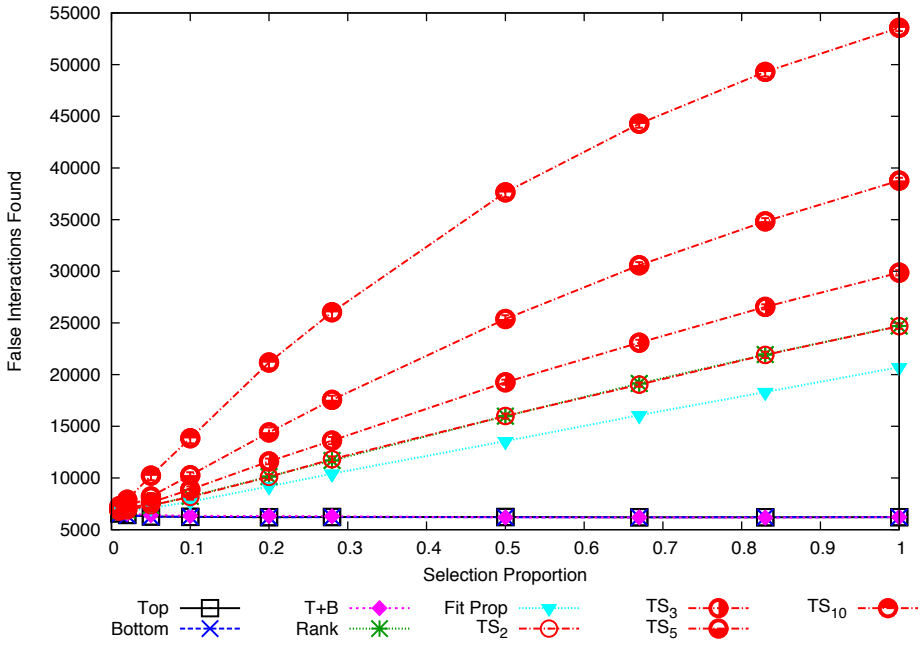

Figure 29: False interactions found for 500 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.

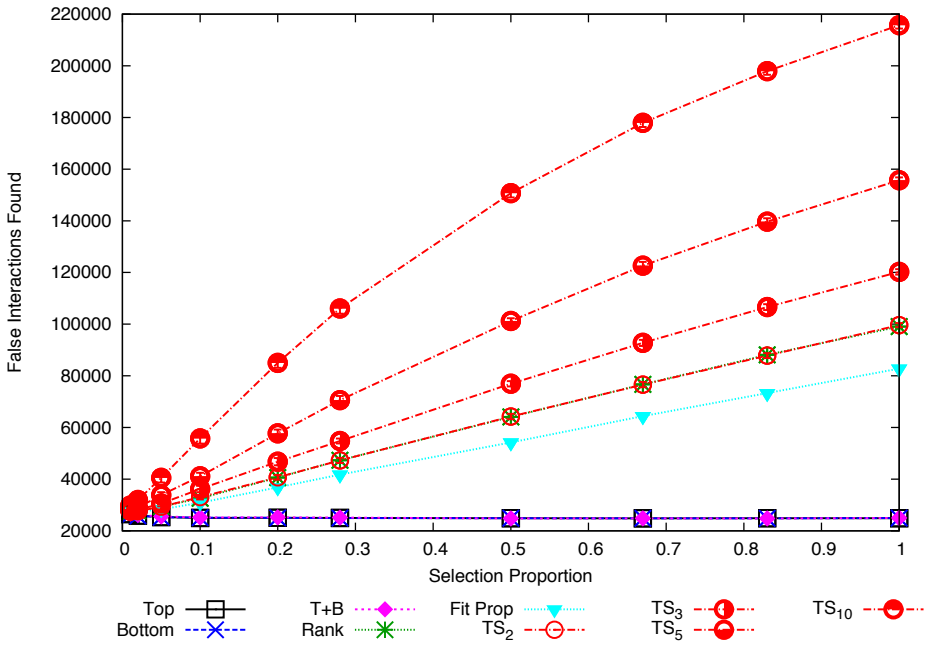

Figure 30: False interactions found for 1000 bit trap-5 with different selection operators and proportions. Error bars show one standard deviation.