EPiC Series in Computing

Volume 81, 2022, Pages 1–11

Proceedings of 11th International Congress on Advanced Applied Informatics

Exploring the limits of an RBSC-based approach in solving the subset selection problem

Kohei Furuya[1], Zeynep Yücel[1], Parisa Supitayakul[1], Akito Monden[1], and Pattara Leelaprute[2]

[1] Department, Faculty, University, City, Country
author1@mail.com, author2@mail.com, author3@mail.com, author4@mail.com
[2] Department, Faculty, University, City, Country
author5@mail.com

Abstract

This study focuses on the *subset selection* problem of computational statistics and deploys the rank-biserial correlation (RBSC) based deck generation algorithm (RBSC-SubGen) [1] in solving it. RBSC-SubGen is originally designed for automatically building a desired number of vocabulary decks (out of a large corpus) with a desired level of word frequency relation, which shares many common aspects with the generic subset selection problem. In this article, we consider applying it not only on word corpora but any set of ranked items and study its resilience against various hyper-parameters, which are not treated in previous studies. Namely, based on simulations we test RBSC-SubGen under various constraints and indicate the vulnerable aspects in terms of rate of saturation, computational cost and accuracy of obtained solution.

1 Introduction and motivation

Most real-life processes involve a certain degree of randomness, the simplest example being rolling a die. The observations obtained from such processes are treated as random variables and represented in terms of stochastic models. The conventional research problem of *Ranking-and-selection* (R&S) of the research field of computational statistics, focuses on such processes involving random factors [2]. Specifically, R&S problem is the problem of choosing the best of two (or sometimes more) processes (or items) which bear a certain degree of randomness, according to a given performance measure. In solving the R&S problems, due to the stochastic nature of the systems, it is desirable to take as many measurements as possible and increase the reliability of decisions. This implies that numerous *physical experiments* need to be carried out. However, in certain cases (e.g. vehicular traffic safety analysis), it may not possible to make physical experiments. In addition, in certain other cases (e.g. pharmacological studies), physical experiments might cost a long time and financial resources.

In that respect, *simulation* is considered to be a suitable tool for the exploration of process performance. Simulations enable extension of the data set, investigation of complex phenomena without the need of analytical solutions as well as assessment of decision paradigms and

algorithms [3]. However, although computer simulations are easier to perform than physical experiments, they are not zero-cost. Namely, since the stochastic models corresponding to the system in focus tend to be highly complex, simulations may cost significant time and computational resources. Therefore, R&S algorithms try to find good solutions in an efficient way, namely, without making an excessive number of simulations.

Most of the R&S algorithms aim choosing the (single) best process out of two or more processes according to a certain performance measure [2]. On the other hand, certain R&S algorithms search for a subset of processes rather than a single one, which are termed as *subset selection* problems [4].

In this article, we focus on the subset selection problem and deploy the RBSC-based deck generation algorithm offered by [1] (henceforth, referred to as `RBSC-SubGen`) to solve it using a variety of hypothetical data sets under diverse constraints. By changing the underlying distribution (e.g. Normal, uniform etc.) and number of samples, we obtain various hypothetical data sets and investigate the resilience of the algorithm against such factors. In addition, by relaxing and tightening the desired specifications of the selected subset, we obtain a variety of constraints and assess the feasibility of the problem, rate of saturation and accuracy of obtained solution. Specifically, we show that the algorithm of [1] can be applied on the subset selection problem. The feasibility of the problem is inferred based on the sizes of the (original) data set and its subsets, while the underlying distribution is observed not to make a significant impact on the results. On the other hand, rate of saturation and accuracy are seen to depend considerably on the pre-defined constraints.

2 Background and related work

Historically, R&S problems are often formulated in agricultural and clinical scenarios (e.g. grain yields, drug treatments) [5]. However, recently their application has expanded to a large domain involding judicial system [6], traffic safety [7], digital photo selection [8] among many others.

As mentioned in Section 1, while most of the R&S algorithms aim choosing the (single) best process [2, 9], certain others search for a subset of processes [10, 11, 4]. Subset selection is essential in evaluating the performance of complex systems using stochastic simulation. Namely, when optimizing system performance with such simulations, the final decision is taken by considering various simulation results. However, since simulations of complex systems are computationally expensive and time-consuming, it is desirable to first eliminate non-competitive designs and then study the remaining ones in detail [12], which can be addressed by subset selection.

Subset selection problems may have various purposes. For instance, if the measurements are known to contain noise, the solution chosen as the single best performing process may not be the *true best solution*, thus choosing a subset and studying them deeper may yield better results [13]. Another potential purpose of subset selection can be post-hoc analysis. For instance, suppose that a group of patients are found to react to a test drug in a certain way based on a set of low-cost medical exams. If one wants to investigate better the pharmacological effects with a reasonable cost, he/she may carry out additional high-cost medical exams on a proper (i.e. diverse as well as representative) subset of patients and achieve reliable results, while keeping cost efficiency. Moreover, a second-stage algorithm or a follow-up simulation of a subsequent process can be tested in conjunction with a subset of solutions (of a preceding process) to achieve more accurate and realistic results (i.e. more robust against measurement noise or variations due to stochasticity).

The chosen subsets are often required to attain the largest or smallest performance measures

and be homogenous [12]. Neveretheless, depending on the application, various other conditions can be imposed as in [1] and [14]. Namely, in [1] the chosen subsets of vocabulary are desired to be uniform within the subset and diverse between the varying subsets, whereas in [14] the chosen subsets are desired to be as diverse and inclusive as possible to satisfy the ethical requirements on fairness.

Various R&S algorithms and subset selections approaches have been proposed including indifference-zone approach (IZ), maximization of the probability of correct selection (PCS), optimal computing budget allocation (OCBA) and expected value of information (EVI) [15], [16], [9]. Since R&S requires pairwise comparison of performance metrics [17], some studies focus on this aspect and try to optimize R&S further by decreasing the number of pairwise comparisons [18].

3 Overview of the algorithm

3.1 Fundamentals of simple difference formula

Essentially, RBSC defines the correlation relating a dichotomous variable and a ranking variable. As an example, consider two track-and-field teams as Team-A and Team-B. For testing the hypothesis that *Team-A is faster than Team-B*, it is an alternative to use RBSC. Suppose that in order to judge the speed of an individual athlete, we use the time it takes him/her to run a track of a certain length. Here, the statement that *Team-A is faster than Team-B* is a dichotomous variable (rated as either `True` or `False`). On the other hand, the time it takes the athletes to run the track can easily be transformed into a ranking variable.

Variable-1:	Team-A is faster than Team-B. $\Rightarrow$ Dichotomous variable (True/False)
Variable-2:	The time it takes the athletes to run a track of certain length. $\Rightarrow$ Transformed into a ranking variable

To evaluate the validity of the above-mentioned hypothesis based on RBSC, initially the times of each pair of athletes from different teams (i.e. one athlete from Team-A and one athlete from Team-B) are compared. Suppose that it takes a particular athlete from Team-A t_a seconds to run the track, where t_a is a real number. Similarly, let a particular athlete from Team-B run the same track in t_b seconds. Let S stand for the number of evidence *supporting* the hypothesis, (i.e. an athlete from Team-A runs faster than an athlete from Team-B),

$$S = 0.5 \cdot \sum_{\forall a,b} \left(\operatorname{sign}\left(t_b - t_a\right) + 1\right),$$

where $\operatorname{sign}(\cdot)$ denotes signum. Similarly, let C represent the number of evidence *contradicting* the hypothesis (i.e. an athlete from Team-B runs faster than an athlete from Team-A),

$$C = 0.5 \cdot \sum_{\forall a,b} \left(\operatorname{sign}\left(t_a - t_b\right) + 1\right).$$

After obtaining the number of evidence supporting and contradicting the hypothesis in this manner, we employ *Simple Difference Formula* proposed by Kerby [19] to compute the RBSC coefficient. According to Kerby [19], the non-parametric correlation equals the simple difference between the proportion of "favorable" and "unfavorable" evidence, where favorable

stands for the pairs supporting the hypothesis and unfavorable for the ones disagreeing with the hypothesis. In explicit terms, RBSC coefficient ρ is computed simply as,

$$\rho = \frac{S - C}{S + C}.$$

The values of ρ are bounded in the range -1 to 1. If the data are all favorable, then the correlation is exactly 1. On the contrary, if the data are all unfavorable, the correlation will be -1, whereas a correlation of 0 indicates equal amount of favorable and unfavorable evidence. In that respect, if $\rho > 0$, then it can be said that the hypothesis holds, and otherwise it does not.

3.2 Outline of RBSC-SubGen

The objective of RBSC-SubGen is to automatically build a desired number of subsets (out of a large set) with a desired level of ranking relation. Since the purpose of our work is not to adapt RBSC-SubGen to a certain data set or to try it with a certain set of constrains, in what follows, we avoid any specifics on the elements of data set and how they are ranked. Suppose that we have a set with a large amount of items (henceforth, referred to as the universal set). Let each item to be associated with a *score*, which can be an objective value (e.g. the time to run a certain track as in the aforementioned example) or a subjective value (e.g. evaluation of a product in an online retail store).

Suppose that our task it is to pick two representative subsets out of the universal set with a desired ranking relation between the items. For assuring that the chosen items are actually representative, they are required to have two properties as (i) *uniformity* and (ii) *diversity*[1]. To assess uniformity and diversity, [1] deploys rank-biserial correlation and updates the randomly chosen subsets in an iterative manner (by adding and removing items, see Alg. 1), until satisfactory levels of uniformity and diversity are attained[2].

Algorithm 1: RBSC-SubGen

Input: Universal set X, size of target subsets S, desired RBSC coefficient ρ^*, tolerable disparity ϵ

Output: Subsets A, B

1 **sample**$(A \subset X, B \subset X \mid |A| = S, |B| = S)$ `// Build potential subsets`
2 $X' = X - A - B$ `// Available items` X'
3 $\rho_{AB} = RBSC(A, B)$
4 **while** $|\rho^* - \rho_{AB}| \geq \epsilon$ **do**
5 $U_A, U_B =$ `GetRequiredUpdate`(A, B, ρ^*, ϵ)
6 $A, X' =$ `UpdateSubset`(A, U_A, X')
7 $B, X' =$ `UpdateSubset`(B, U_B, X')
8 $\rho_{AB} = RBSC(A, B)$
9 **return** A, B

The most essential step of RBSC-SubGen is this iterative update (see Lines 4-8 of Alg. 1). Initially, the current state of the subsets is assessed and the sort of necessary update is de-

[1] Uniformity refers to a state, in which the scores of the items in the same subset are uniform enough so that they can be grouped together. In addition, diversity refers to a state, in which the scores of the items in different decks are diverse enough so that they can be grouped in different decks.

[2] In case of [1], the large set is a lexicon, the subsets are vocabulary decks and the scores are the number of occurrences of the words collected from Wiktionary.

termined as illustrated in Alg. 2. Suppose that between the subsets A and B, the one with lower scores (on the average) is A. Further suppose that ρ_{AB} is lower than the desired value ρ^* than a maximum permissible disparity ϵ, i.e. $\rho^* - \rho_{AB} > \epsilon$. In this case, A has to be updated such that an item with a relatively low score needs to be appended to it and an item with a relatively high score needs to be returned from A to the universal set. Moreover, B has to be updated such that an item with a relatively high score needs to be appended to it and an item with a relatively low score needs to be returned from B to the universal set. We code the aforementioned update on A and B with $U_A = -1$ and $U_B = 1$. If ρ_{AB} is higher than the desired value ρ^* beyond ϵ (i.e. $\rho_{AB} - \rho^* > \epsilon$), the updates will be opposite. However, if ρ_{AB} is between an interval of $\pm\epsilon$ around ρ^* ($|\rho_{AB} - \rho^*| \leq \epsilon$), no action is necessary (coded as $U_A = U_B = 0$).

Algorithm 2: `GetRequiredUpdate`

Input: Subsets A and B, desired RBSC coefficient ρ^*, permissible disparity ϵ
Output: Required updates U_A and U_B

$U_A, U_B = 0, 0$
$\rho_{AB} = RBSC(A, B)$
if $\rho_{AB} < \rho^* - \epsilon$ **then** U_A, U_B = -1, 1
else if $\rho_{AB} > \rho^* + \epsilon$ **then** U_A, U_B = 1, -1
return U_A, U_B

For updating a subset, the procedure illustrated in Alg. 3 is followed. At every iteration, a single item is removed from the subset and returned to the universal set, while a single item from the universal set is appended to the subset. The item to be added or removed is chosen such that it provides a change on ρ_{AB} in the desired direction (i.e. increasing or decreasing). The random sampling of an item a with a high score from the subset A is represented with **sample**$(a \in A \mid s_a > m_A)$, where s_a denotes the score of the item a and m_A is the median score of the items in A. Obviously, an arbitrary sample a may not satisfy $s_a > m_A$, and the sampling operation may need to be repeated several times. In addition, in order to avoid extremely long running times (or divergence), a maximum number of iterations N_{max} needs to be defined. After repeating the update operation illustrated in Alg. 3 for the two subsets A and B (see Lines 6, 7 in Alg. 1), the new value of RBSC coefficient is computed. Should it turn out be within the satisfactory range ($|\rho_{AB} - \rho^*| < \epsilon$), the updates are terminated and the subsets are returned.

Algorithm 3: `UpdateSubset`

Input: Subset A, required update U_A, available items in the universal set X'
Output: Updated subset A, available items in the universal set X'

if $U_A = -1$ **then** `// Decrease scores in` A
  `/* Pick a high-score item` a `from` A `and a low-score item` x `from` X' `*/`
  sample$(a \in A, x \in X' \mid s_a > m_A, s_x < m_A)$
else if $U_A = 1$ **then** `// Increase scores in` A
  `/* Pick a low-score item` a `from` A `and a high-score item` x `from` X' `*/`
  sample$(a \in A, x \in X' \mid s_a < m_A, s_x > m_A))$
$A = A \bigcup \{x\} \setminus \{a\}$ `/* Move` a `from` A `to` X' `and` x `from` X' `to` A `*/`
$X' = X' \bigcup \{a\} \setminus \{x\}$

4 Investigation of convergence properties of the algorithm

In this section, we examine RBSC-SubGen in relation to a set of hyper-parameters and elaborate on quantification of its convergence properties based on a set of performance measures.

4.1 Hyper-parameters

The hyper-parameters which are considered to have a direct impact on the operation of the algorithm are presented in Table 1. In what follows, we give a brief description of each hyper-parameter and discuss the anticipated impacts against changes on its values.

The desired value of the RBSC coefficient between two subsets is denoted with ρ^*. Obviously, as ρ^* increases there will be a larger margin between the scores of the items in the subsets. Thus, higher values of ρ^* indicate more challenging problems, which are likely to terminate after a larger number of iterations. If ρ^* is significantly high, the algorithm is likely to get saturated, i.e. reach the maximum number of iterations before achieving convergence.

Table 1: Hyper-parameters of the algorithm.

Variable	Description	Min	Max	Step size	Default
ρ^*	Desired value of RBSC coefficient	0.3	0.7	0.04	0.5
ϵ	Maximum permissible disparity on ρ^*	0.05	0.15	0.01	0.1
L	Size of the universal set X	100	900	50	500
S	Size of subsets A, B	100	500	20	300

In most real-world problems, the universal (i.e. source) set (collected through physical or simulated experiments) has a limited size. This implies that there may not always be enough freedom (in choosing the scores) for achieving the exact value of ρ^*. In that case, it is necessary to define a maximum permissible value ϵ for the disparity on ρ^*, $\Delta\rho = |\rho^* - \rho|$. Provided that ρ is in the range $[\rho^* - \epsilon, \rho^* + \epsilon,]$, or equivalently $\Delta\rho \leq \epsilon$, it is considered to be sufficiently close to the desired value. Clearly, if the maximum permissible disparity ϵ is low, a more accurate solution is desired, which may require a larger number of iterations, or may even result in a failure in achieving convergence.

Since the universal set is likely to have a limited size, it is worth devoting a hyper-parameter to account for this challenge. In this study, the size of universal set is denoted with L. If L is large, the algorithm is expected to have sufficient freedom in choosing the scores and to be likely to converge, provided that the other hyper-parameters are not too strict. Nevertheless, if L is too small, even if the other hyper-parameters are not too strict, the algorithm is likely to terminate without achieving convergence.

In this work, RBSC-SubGen [1] is confined to generate *a pair* of target subsets A and B, where the size of each subset is denoted with S[3]. Here, it is important to note that if L is lower than $2 \cdot S$, then there is no solution, since the problem is inherently not workable. In addition, although an increase on S has a limiting effect on the freedom of choosing the scores, provided that L is large enough, this effect can be expected to diminish.

It is also interesting that an increase on S can somewhat make the effect of individual scores less significant. Namely, since the number of possible pairs is proportional to S^2, as S increases,

[3]In practice, RBSC-SubGen can be executed to generate any desired number of subsets. Here, we consider generating a pair of (i.e. 2) subsets as an initial step in investigating its feasibility.

the number of possible pairs increase even with a larger margin[4]. In that respect, the potential impact of a single modification decreases with increasing S.

4.2 Several other specifics of testing

RBSC-SubGen is tested with several sets of hyper-parameter values. Specifically, we consider the hyper-parameters to take the values presented in Table 1. In this table, Min and Max denote the lowest and highest values assumed by each hyper-parameter, where certain intermediate values at given step sizes are also used in testing to observe the evolution of performance. In addition, in order to visualize the pair-wise relations (i.e. the interplay of two hyper-parameters), the remaining two hyper-parameters are set to certain default values. For instance, while examining the interaction of universal set size L and subset size S, desired value of RBSC coefficient ρ^* and disparity on the desired RBSC coefficient ϵ are fixed at 0.5 an 0.1, respectively. In specific, each default value is set to the median of the corresponding range.

Note that provided that $L > 2 \cdot S$, there is always some item which is not yet assigned to any of the subsets A, B. If ρ_{AB} does not satisfy $|\rho_{AB} - \rho^*| \leq \epsilon$, then such items can be used to update the subsets. However, in certain cases (e.g. too little freedom in sampling or too little impact of modifying a single item), the algorithm may go into an infinite loop with no avail, if one does not define a limit on the number of such iterations (see Line-4 of Alg. 1). Therefore, it is necessary to limit the maximum number of iterations for identifying such bottlenecks. Here, the maximum of iterations is denoted with M. If the algorithm reaches M iterations without achieving an RBSC coefficient within the range $[\rho^* - \epsilon, \rho^* + \epsilon]$, it is said to get *saturated*.

In addition, regarding each set of hyper-parameters the algorithm is executed I times, such that at each execution a fresh universal set is generated randomly. In addition, concerning each execution we compute the performance indicators described in Section 4.3 and report the relating statistics in Section 5.

4.3 Quantification of performance

For assessing the performance of RBSC-SubGen, several indicators are considered such as rate of saturation, number of iterations until convergence and disparity on desired RBSC coefficient.

As mentioned in Section 4.2, if the algorithm reaches M iterations without achieving an RBSC coefficient within the range $[\rho^* - \epsilon, \rho^* + \epsilon]$, it is said to get *saturated*. One of the performance indicators in assessing the performance of the algorithm is the *rate of saturation* denoted with β_o. Namely, the ratio of the number of saturated runs to the number of all runs is registered as rate of saturation. This value indicates how often RBSC-SubGen fails to build the subsets.

If the algorithm achieves an RBSC coefficient satisfying $\Delta\rho \leq \epsilon$ and in less than M iterations, it is said to *converge*. Here, the number of iterations until convergence denoted with M_o is considered as another performance indicator. This value indicates how quickly RBSC-SubGen builds the subsets.

Note that if the algorithm gets saturated, it does not converge and thus the number of iterations (i.e. M) is not considered in computing M_o. Moreover, in the case that the algorithm gets constantly saturated for a certain set of hyper-parameters, M_o is not available.

As explained in Section 4.1, ϵ defines the maximum permissible disparity on ρ^*. Here also the observed value of disparity $\Delta\rho$ is considered as a performance indicator. Note that the observed disparity on the desired value of RBSC $\Delta\rho$ ideally depends strictly on ϵ[5].

[4] In other words, the modification of an individual score may affect at most S cases.

[5] In that respect, the reason for computing $\Delta\rho$ is for making sure that the number of epochs I is sufficiently

5 Results and discussion

In this section, the observations relating each performance indicator are illustrated and a discussion is provided on the sensitivity of the algorithm to the modifications on their values.

5.1 Results and discussion relating rate of saturation

Rate of saturation is considered to be the most challenging aspect of the performance of RBSC-SubGen. Namely, saturation implies that an RBSC coefficient cannot be achieved within the permissible range and the algorithm is considered to fail. It can be seen in Figure 1 that such a serious case occurs most often when the hyper-parameter pair (ρ^*, S) pose a challenge. Specifically, if both the desired value of RBSC coefficient ρ^* and the size of target subsets S are large, then there is risk that the algorithm gets saturated. One may see that large values of ρ^* pose a risk of saturation also when they are coupled with small values of ϵ and small values of L, where the algorithm is observed to be more sensitive to the former (see Figure 1-(d)) as compared to the latter (see Figure 1-(b)). The same holds also for S. In other words, when large values of S are coupled with small values of ϵ or small values of L, a risk of saturation emerges. Nevertheless, comparing ϵ and L, one may notice that the risk is more serious for ϵ than L. As for the pair (ϵ, L), it can be seen in Figure 1 that the risk of saturation is not serious and the algorithm is likely to yield the outcome satisfying the desired properties.

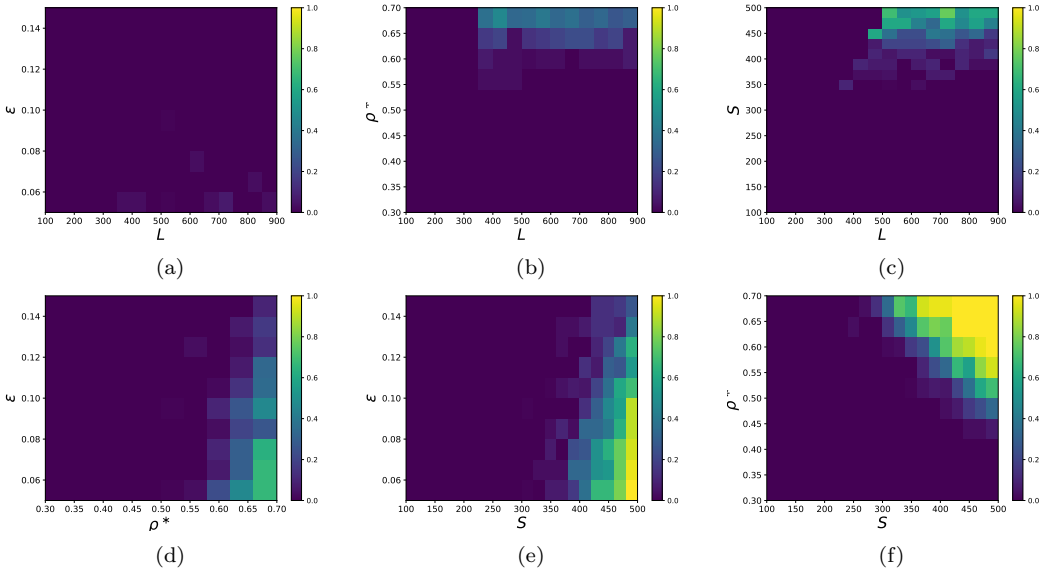

Figure 1: The effect of each possible hyper-parameters pair on rate of saturation β_o.

high. In other words, if it can be observed that $\Delta\rho$ depends solely on ϵ, it can be claimed that the values β_o and M_o are likely to grasp the general characteristics and are reliable, and additional test runs are not necessary.

5.2 Results and discussion relating number of iterations

Subsequently, we examine the number of iterations M_o that it takes RBSC-SubGen to terminate successfully. We report our observations in Figure 2, where the dark blue regions signify either (i) the cases where the problem is not feasible (i.e. $L < 2 \cdot S$) or (ii) that all I runs are found to be saturated. If (i) is the case, there exists no solution, but if (ii) is the case one may try to improve the algorithm. In particular, the dark blue regions in Figures 2-(a), (b), (c) arise due to an infeasibility. On the other hand, certain sets of hyper-parameter values seen in Figures 2-(e), (f), end up in saturation since it is not possible to find a solution within computation limits.

Next, omitting the aforementioned cases (i) and (ii), we focus on successful executions. Here, it is interesting that although the hyper-parameter pair of (ρ^*, S) has the largest number of saturations, a higher number of iterations are required to solve most problems relating the hyper-parameter pair of (ϵ, S). This observation is somewhat valid also for the hyper-parameter pair of (ϵ, ρ^*). Namely, in the overall, the number of iterations are higher for most (ϵ, S) and (ϵ, ρ^*) pairs, although saturation is not as common as in the case of (ρ^*, S).

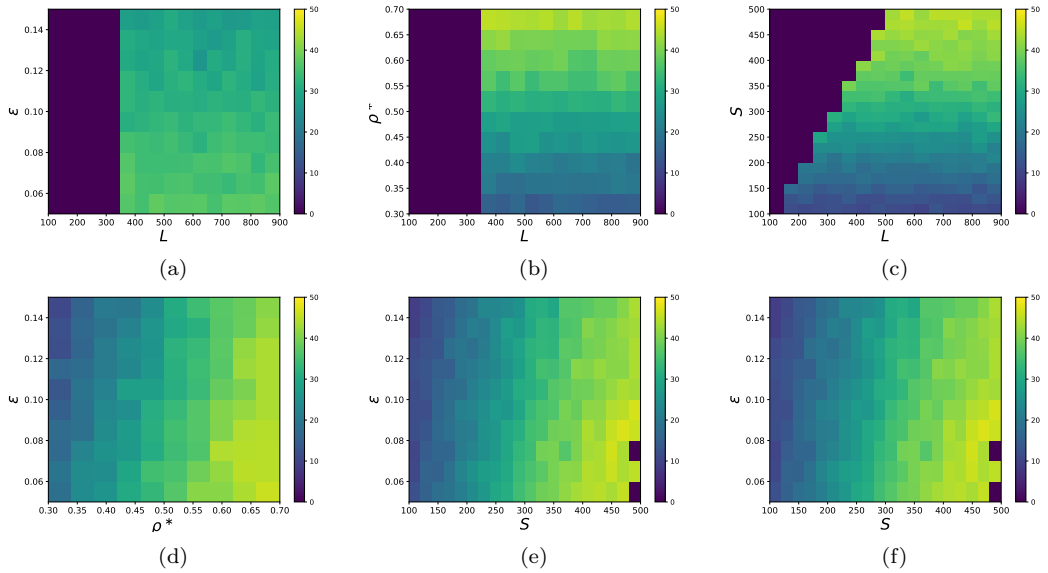

Figure 2: The effect of each possible hyper-parameters pair on number of iterations M_o.

5.3 Results and discussion relating disparity on desired RBSC

We also investigated the evolution of disparity $\Delta\rho$ on the desired RBSC coefficient ρ^* relating to the same six hyper-parameters pairs. It is observed that as long as ϵ is one of the hyper-parameters under investigation, $\Delta\rho$ depends solely on it and the other hyper-parameter under investigation does not introduce a prominent effect on the rate of saturation or the number of iterations necessary to find a solution. This is not surprising since termination of iterations is decided based on ϵ (see Line-4 of Alg. 1).

6 Conclusion

We considered RBSC-SubGen, which is originally designed for automatically building a desired number of vocabulary decks (out of a large corpus) with a desired level of word frequency relation. Exploiting the fact that this objective shares many common aspects with the generic subset selection problem, we tried RBSC-SubGen in generating subsets out of universal sets coming from different underlying distributions [6] and of different size. We also imposed varying constraints on subset size, desired ranking relation and permissible disparity. Our results indicate that RBSC-SubGen can be used in subset selection, provided that it is formulated as a ranking relation. In addition, RBSC-SubGen is found to be sensitive to subset size S, followed by desired RBSC coefficient ρ^*, permissible disparity ϵ and, finally, universal set size L.

Acknowledgements

The codes developed for the analysis reported in this article are released in our repository [20].

References

[1] Z. Yücel, P. Supitayakul, A. Monden, and P. Leelaprute, "An algorithm for automatic collation of vocabulary decks based on word frequency," *IEICE Tran. Information and Systems*, vol. 103, no. 8, pp. 1865–1874, 2020.

[2] L. J. Hong, W. Fan, and J. Luo, "Review on ranking and selection: A new perspective," *arXiv preprint arXiv:2008.00249*, 2020.

[3] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Tran. Software Engineering*, vol. 29, no. 11, pp. 985–995, 2003.

[4] D. J. Eckman, M. Plumlee, and B. L. Nelson, "Revisiting subset selection," in *Proc. Winter Simulation Conf.*, pp. 2972–2983, IEEE, 2020.

[5] S. S. Gupta, "On some multiple decision (selection and ranking) rules," *Technometrics*, vol. 7, no. 2, pp. 225–245, 1965.

[6] L. Huang, J. Wei, and E. Celis, "Towards just, fair and interpretable methods for judicial subset selection," in *Proc. AAAI/ACM Conf. AI, Ethics, and Society*, pp. 293–299, 2020.

[7] G. C. McDonald, "Applications of subset selection procedures and Bayesian ranking methods in analysis of traffic fatality data," *Computational Statistics*, vol. 8, no. 6, pp. 222–237, 2016.

[8] C.-H. Yeh, B. A. Barsky, and M. Ouhyoung, "Personalized photograph ranking and selection system considering positive and negative user feedback," *ACM Tran. Multimedia Computing, Communications, and Applications*, vol. 10, no. 4, pp. 1–20, 2014.

[9] C.-H. Chen, S. E. Chick, L. H. Lee, and N. A. Pujowidianto, "Ranking and selection: Efficient simulation budget allocation," *Handbook of Simulation Optimization*, pp. 45–80, 2015.

[10] S. H. Choi and T. G. Kim, "A heuristic approach for selecting best-subset including ranking within the subset," *IEEE Tran. Systems, Man, and Cybernetics-A*, vol. 50, no. 10, pp. 3852–3862, 2018.

[11] M. H. Alrefaei and M. Almomani, "Subset selection of best simulated systems," *Journal of the Franklin Institute*, vol. 344, no. 5, pp. 495–506, 2007.

[12] Y. Wang, L. Luangkesorn, and L. J. Shuman, "Best-subset selection procedure," in *Proc. Winter Simulation Conf.*, pp. 4310–4318, IEEE, 2011.

[13] S. Gao, H. Xiao, E. Zhou, and W. Chen, "Robust ranking and selection with optimal computing budget allocation," *Automatica*, vol. 81, pp. 30–36, 2017.

[6] The figures presented in Section 5 are for standard Normal distribution. We examined the results concerning other distributions and did not observe any significant differences.

[14] M. Mitchell, D. Baker, N. Moorosi, E. Denton, B. Hutchinson, A. Hanna, T. Gebru, and J. Morgenstern, "Diversity and inclusion metrics in subset selection," in *Proc. AAAI/ACM Conf. AI, Ethics, and Society*, pp. 117–123, 2020.

[15] S. Gao and W. Chen, "A note on the subset selection for simulation optimization," in *Proc. Winter Simulation Conf.*, pp. 3768–3776, IEEE, 2015.

[16] C.-H. Chen, D. He, M. Fu, and L. H. Lee, "Efficient simulation budget allocation for selecting an optimal subset," *INFORMS Journal on Computing*, vol. 20, no. 4, pp. 579–595, 2008.

[17] M. J. Groves, *Efficient Pairwise Information Collection for Subset Selection.* PhD thesis, University of Warwick, 2020.

[18] L. J. Hong, J. Luo, and Y. Zhong, "Speeding up pairwise comparisons for large scale ranking and selection," in *Proc. Winter Simulation Conf.*, pp. 749–757, IEEE, 2016.

[19] D. S. Kerby, "The simple difference formula: An approach to teaching nonparametric correlation," *Comprehensive Psychology*, vol. 3, pp. 11–IT, 2014.

[20] Z. Yücel, "Solving the subset generation problem with RBSC-SubGen." https://github.com/yucelzeynep/Subset-generation-with-RBSC-SubGen, 2021. [Accessed 2021-10-18].