Maximizing Diversity When Partitioning a Set into Subsets of Equal or Near Equal Cardinality*

Grantham Gollier†

Abstract. Partitioning a set of discrete elements (such as people) into subsets of equal, or approximately equal, cardinality with a maximized diversity is a common problem. This papers describes an ecological equation known as biodiversity as a metric for use with Simulated Annealing. In order to judge effectiveness, we compared the above approach with a random match based on the score they received using the aforementioned biodiversity metric on multiple datasets. The Simulated Annealing algorithm, although computationally more intensive, resulted in significantly more diverse subsets.

Key words. combinatorial optimization, Simpson's Diversity Index, Simulated Annealing

AMS subject classifications. 68R05

1. Introduction.

The problem with which this research concerns itself relates to the following situation: a set of students will be partitioned into subsets of approximately equal cardinality. Each student has a list of qualities: school year, gender, mathematical proficiency, writing proficiency, interest in sports, etc. The goal is to create subsets of approximately equal (and known) cardinality so that the diversity of each subset is maximized. This is an NP-hard combinatorial optimization problem.

Simulated Annealing (SA) is a computationally efficient and common approach for solving combinatorial optimization problems [2]. Simulated Annealing, in order to be effective, requires a metric to be optimized. Simpson's Diversity Index has long been used to quantify diversity in biological populations; we used this quantity as a metric for the diversity of each subset.

This paper is organized as follows. Section 2 reviews the SA algorithm and Simpson's Diversity Index. Our new algorithm is presented in section 3. Results for two different data sets are presented in section 4. And section 5 provides a summary of the performance of the algorithm and suggests future lines of work.

2. Background.

In this section, we discuss the SA algorithm and Simpson's Diversity Index. We show how to use SA to create subsets with high diversity using Simpson's Diversity Index in section 3.

2.1. Simulated Annealing Algorithm.

SA is a robust optimization algorithm that is based on the physical model of how metals cool [5]. In each iteration, the SA algorithm has a current solution and a proposed solution, which may be worse. Rather than accepting the proposed solution if it is better and rejecting the proposed solution if it is worse, SA accepts or rejects the proposed solution according to a stochastic formula. If a liquid metal cools slowly, its atoms form a pure crystalline structure, corresponding to a lower energy state. If a liquid metal cools quickly, its atoms do not have time to arrange themselves optimally, so the metal

*Submitted to the editors DATE.

†University of Kansas, Lawrence, KS (ggollier@ku.edu).

freezes in a higher energy state. SA uses this model of cooling where the objective function represents the energy state. At each iteration, SA will usually choose a lower energy state; sometimes, though, it will choose a higher energy state

Pseudo-code for SA is listed in Algorithm 1, where s_0 is the initial state solution, T_0 is the initial temperature, T_{min} is the minimum temperature, and k_{max} is the maximum possible number of iterations at each selected solution. The larger k_{max} the more likely and optimally solution will be found, but the compute time will rise almost directly with it. The parameter $\alpha \in (0, 1)$ determines how quickly the system "cools"; the smaller the alpha, the quicker the convergence. However, a larger alpha permits SA to explore the parameter space more and possibly generate a better solution. The function AP calculates the acceptance probability, that is, a value in the range [0,1] that represents the probability of changing solutions. The random function, which outputs a random number in the range [0,1], keeps the algorithm from terminating at a local maximum, especially in early iterations.

Algorithm 1 Pseudo-code for Simulated Annealing Algorithm

```
Let s = s_0
Let h = objective(s)
Let T = T_0
while T ≥ T_min do
    for k = 0 through k_max do
        s_temp = neighbor(s)
        h_temp = objective(s_temp)
        if AP(h_temp, h, T) > random(0, 1) then
            s = s_temp
            h = h_temp
            break
        end if
    end for
    T = T * α
end while
Return: s
```

2.2. Biodiversity. In Ecology it is often useful to quantify the biodiversity of a habitat. Biodiversity is the the variety of organisms in a specific habitat or system. There are many mathematical models for calculating this biodiversity in a given area. Any method for calculating biodiversity is dependent on the richness and evenness of the population.

The number of species per sample is a way to measure richness. The more species one sample has the richer it is. However, species richness on its own does not take into account the number of organisms in each species. There could be one species with 1 organism and another with 1000, and they would be weighted equally. For this reason, richness alone is not a satisfactory measure of biodiversity.

Evenness is the measure of relative abundance of different species making up the richness of a sample. A sample is considered more even if the difference between the number of individuals in each species is less. For example, consider the samples in Table 1, Sample 2 is

Table 1
Biodiversity Example Comparing Two Samples of Two Species

Species	Sample 1	Sample 2
Daisy	90	49
Sunflower	10	51
Total	100	100

much more even than Sample 1 because the difference in number of organisms between species is 2 in Sample 2 as opposed to 80 in Sample 1.

2.3. Simpson's Diversity Index. Separate numbers for richness and evenness can not be used as the objective function in SA. Instead, we seek a single number that quantifies diversity. As a species' richness and evenness increase, so does its diversity. Simpson's Diversity Index takes both richness and evenness into account in order to quantify biodiversity [4]. The Simpson's Index of Diversity (SID) of a system, D is defined in (1) where n is the total number of organisms of a particular species and N is the total number of organisms in the system.

$$D = 1 - \sum (\frac{n}{N})^2 \tag{1}$$

Using SID, the biodiversity is given as a number in the range [0, 1] where 1 represents infinite diversity and 0 represents no diversity. Using SID, the index represents the probability that two organisms taken from the population are of different species. In the order to maximize diversity of the partitions, this SID is what needs to be maximized.

3. Theory. Our implementation of SA is similar to that which is outlined in section 2.1. We use a modification of the SID equation which is defined in (2) such that a_i is the SID for the i^{th} quantifier and n is the number of elements in the subset with the specified quantifier value and N is the cardinality of that subset, and w_i is the weight for the current quantifier (n.b. all weights in each subset must add to 1 and quantifiers must be discrete measurements). The objective function, for the current subset g, is then calculated by summing these weighted SID's. The set $\mathbb{O}$ contains all of these subsets' objective values, which are then averaged to calculate the overall objective value for the current solution, s.

$$\begin{aligned} a_i &= w_i * \left(1 - \sum (\frac{n}{N})^2\right) \\ \mathbb{O} &= \left\{o_g : o_g = \sum a\right\} \\ objective(s) &= average(\mathbb{O}) \end{aligned} \tag{2}$$

We define the acceptance probability function, AP, in (3). In this equation, h is the objective value of the currently selected solution, h_{temp} is the objective value of the neighboring solution, which is to be compared to the current solution, and T is the current temperature.

$$AP(h_{temp}, h, T) = e^{\frac{h_{temp} - h}{T}} \tag{3}$$

SA also requires a method to generate similar solutions, given one solution. These similar solutions are called neighbors. A major part of SA involves comparing the SID of these neighbors to the current solution, as to find the maximum SID. If neighbors are calculated incorrectly then SA will fail. Neighbors are defined by converting the solution into an n-dimensional vector, where n is the number of elements in the original set and each element of the vector represents the specific subset each element was partitioned into. Then, a $\beta \in (0, n]$ number of elements are randomly rearranged with each other to create a similar, but slightly different vector compared to the initial one. This resultant vector is the neighbor for the SA algorithm. A smaller β value will result in a more conservative algorithm, resulting in solutions that are more similar to the initial one. A larger β, conversely, will result in an algorithm where solutions share little in common and a large enough $/beta$ would result in a modified random search algorithm. Experimentally, we have found a β of approximately a quarter the cardinatily of the set to be optimal.

4. Results.

In the following subsections, there are results for two different problems. One concerning students, who need to be partitioned as diversely as possible into lunch table assignments with known gender and age. This problem will be referred to as the "Lunch Table Problem" and is discussed in section 4.1. The other problem, a toy problem, concerns a business with one hundred employees of known age, gender, mathematical proficiency, programming proficiency, etc. It has been shown that there is a correlation between diversity and financial performance [1, 3]. The manager would like to partition the employees into ten teams of ten so that the teams are diverse and prevent overlapping skills. This problem will be called the "Toy Problem" and is discussed in section 4.2.

4.1. Lunch Table Problem.

With approximately two hundred students, there are 200! or $\sim 3.9 * 10^{374}$ ways to arrange the students. Therefore, it is not possible to fully calculate every permutation of students so that the optimal result can be found. Also, the only information available for every student is grade and gender. Gender alone is not a great optimization feature because of its binary nature, so we used both gender and grade.

The subsets created with SA were significantly more diverse (according to SID) than those generated randomly. This distribution is expressed in Figure 1 and the means and compute time in Table 2.

We achieved these results using a random solution for s_0, a T_0 of 1, a T_{min} of 0.001, an α of 0.9, a β for use with the neighbor function of 45, and a k_{max} of 300. We set the weights (w_1 and w_2) to 0.5; that is to optimize grade and gender equally.

Table 2
SID Score and Compute Time for Lunch Table Problem

Method	Random	Simulated Annealing (SID)
Average SID Score (1000 runs)	0.0002	0.0016
Average Compute Time (in seconds, 1000 runs)	0.0052	168.78

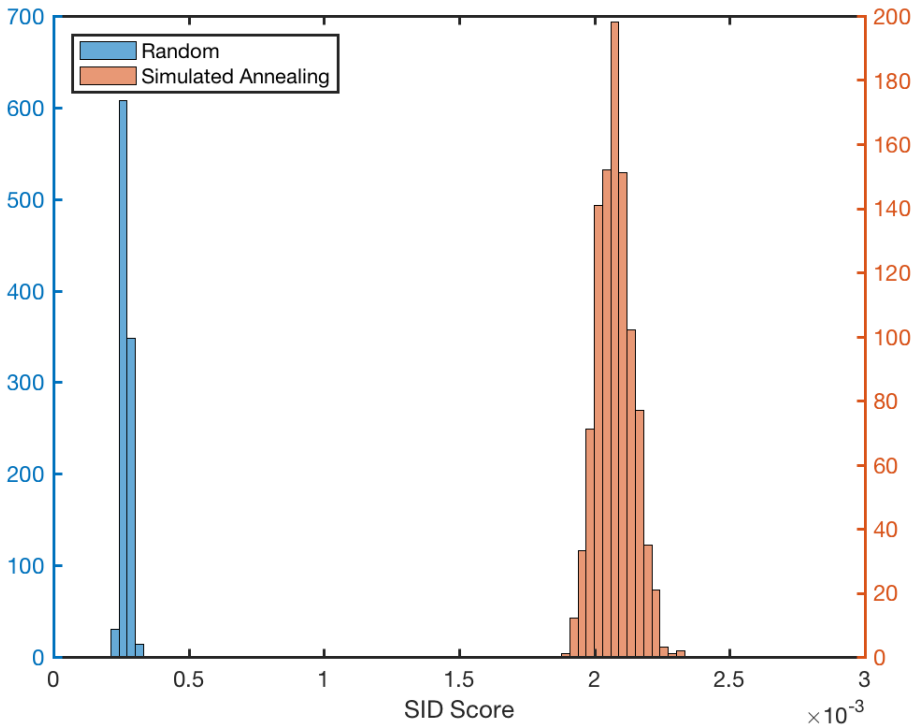

Figure 1. *Distribution of SID Score vs Method of Generation for Lunch Table Problem*

4.2. Toy Problem. In this problem, the manager must partition one hundred employees into 10 subsets of equal cardinality. Each employee has three random integer quantifier values varying from 1 to 10. These values were set once and used for each run so to test the algorithm on the same random data set. Let's call these quantifiers programming proficiency, language proficiency, and leadership skills.[1]

Just like with the Lunch Table Problem, the SA generated subsets were significantly more diverse (according to SID) than their random counterparts. This distribution is expressed in Figure 2.

We achieved these results using a random solution for s_0, a T_0 of 1, a T_{min} of 0.001, an α of 0.9, a β value for use with the neighbor function of 25, and a k_{max} of 300. We set the weights w_1 and w_2 to 0.333 and w_3 to 0.334. The goal was to weigh all quantifiers equally,

[1]This data set can be found at https://people.eecs.ku.edu/~g750g706/toy-set.csv

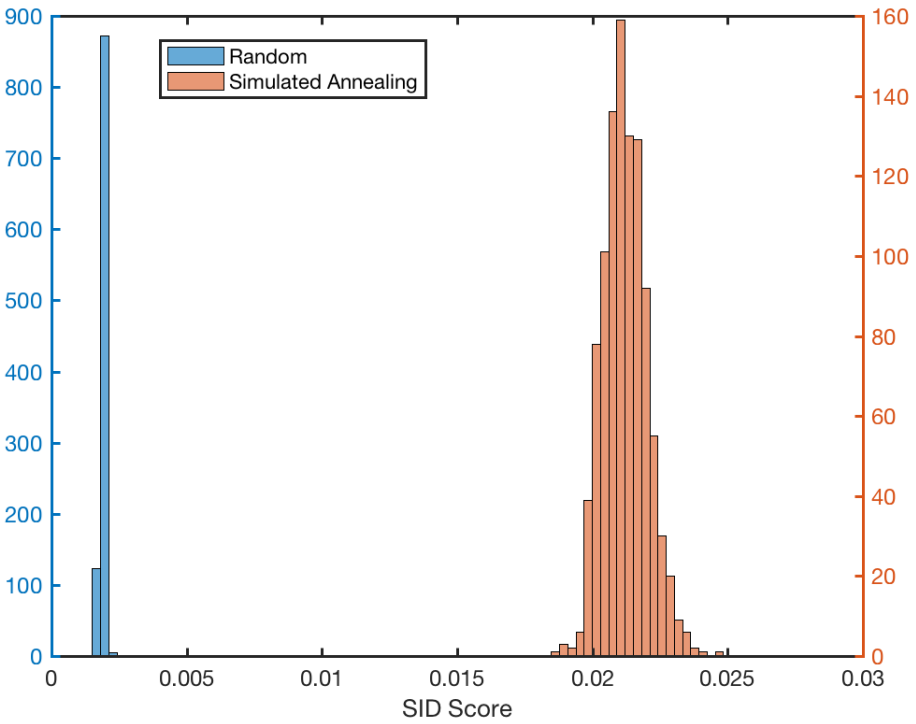

Figure 2. *Distribution of SID Score vs Method of Generation for Toy Problem*

Table 3
SID Score and Compute Time for Toy Problem

Method	Random	Simulated Annealing (SID)
Average SID Score (1000 runs)	0.0019	0.0212
Average Compute Time (in seconds, 1000 runs)	0.0039	83.46

but there was a slight preference to quantifier 3: leadership skills.

5. Conclusion. In the future, for the Lunch Table Problem, currently repeating tables are only considered in the initial random seed, which means that this feature may not be present in the outputted solution. Therefore, in the future, table repetition needs to be considered as a negative weight in the metric. Also, the current average compute time of 168 seconds could also be improved by optimizing the current implementation of the SA algorithm, such as decreasing the k_{max} value since this value almost directly correlates to compute time. In the Toy Problem, again compute time is much larger than what would be desirable and a more involved tuning of k_{max} could improve this compute time.

6. Acknowledgements. The author would like to thank Nicholas Dwork for his continued help throughout the research and development process and his tireless effort revising the manuscript, and Christopher Bryan for his helpful knowledge of Simpson's Diversity Index.

REFERENCES

[1] V. Hunt, D. Layton, and S. Prince, *Diversity Matters*, tech. report, McKinsey and Company, February 2016.
[2] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, Springer Science & Business Media, 2013.
[3] M. A. Neale, K. A. Jehn, and G. B. Northcraft, *Why Differences Make a Difference: A Field Study of Diversity, Conflict, and Performance in Workgroups*, Administrative Science Quarterly, 44 (1999), pp. 741–763.
[4] E. H. Simpson, *Measurement of Diversity*, Nature, (1949).
[5] B. Suman and P. Kumar, *A Survey of Simulated Annealing as a Tool for Single and Multiobjective Optimization*, Journal of the Operational Research Society, 57 (2006), pp. 1143–1160.