Search Space Restriction for UCSP in Genetic Algorithms via a Novel Random-key Decoder

Ian Smith
Computer Science Department
Cranbrook Kingswood Upper School
Bloomfield Hills, MI 48303, USA
isilversmith810@gmail.com

George Pappas, Ph.D.
Electrical and Computer Engineering Department
Lawrence Technological University
Southfield, MI 48075, USA
gpappas@ltu.edu

Abstract

The University Class Scheduling Problem (UCSP), or the process of building a class schedule for a university, has been a difficult problem in computer science due to its combinatorial complexity and multi-objective nature. As a result, several heuristic-based approaches have been tested, one being the genetic algorithm (GA). The application of GAs to this problem has presented its own difficulties due to the generation of infeasible solutions. Thus, this article presents novel methods of decoding random-key genetic representations such that only feasible solutions are generated. It is then demonstrated empirically that these modifications still constitute a viable algorithm, and analytically that these changes affect no additional bias on the otherwise unmodified GA. Going forward, similar, more generalized methods can be developed and applied to a wider range of problems.

Keywords
schedule formation, genetic algorithm, optimization, random-key

1. Introduction

Scheduling problems, or timetabling problems, essentially involve the distribution of limited resources over a finite space. This article focuses on a specific variation of this problem called the University Class Scheduling Problem, which is often considered to be a combinatorial problem and known to be NP-hard. UCSP can be broken down into hard constraints (room capacity, one class per room per period, etc.), which must all be satisfied for a solution to be viable, and soft constraints (teacher preferences, room utilization, etc.), which should be optimized to create "good" solutions.

The presence of both hard and soft constraints has led to the development of several approaches towards optimizing UCSP, some of which are exact and others heuristical. Some exact approaches include brute-force search and integer linear programming. Integer linear programming takes a selection of variables and optimizes them based on a mathematical model while keeping the inputs bounded by a set of constraints. Both of these approaches suffer from combinatorial intractability as the search space increases. Due to these issues leading to a lack of satisfactory solutions, approximate methods were developed. Some heuristical approaches include genetic algorithms and particle swarm optimizers, as well as tabu search which uses meta-heuristics.

This article focuses on the application of genetic algorithms in particular. GAs are one of the most powerful heuristic optimizers since they use random noise to explore a problem's search space, meaning no additional information about the function being optimized is required. This is blindness towards the loss function is not required; local optimizations have been employed for some problems such as the traveling-salesman problem.

GAs operate on a population of individuals, where each individual contains one or more "chromosomes" which encode an input into a loss function. These chromosomes are usually in the form of bit strings or real-valued vectors. These abstract representations are necessary for consistent methods of generating new individuals. A translator is almost always required to convert the abstract representation into one which can be evaluated by a loss function. This article demonstrates how this translator can be modified to satisfy the constraints of UCSP by operating on a set of real-valued vectors, often termed "random-keys."

2. Literature Review

As mentioned in the introduction, several papers have been published which take exact approaches through the method of integer linear programming. Samiuddin et al. used this method with novel adjustments to improve performance and computational cost while solving UCSP. The inspiration for the proposed method of embedding hard constraints in the chromosome decoder came from their problem decomposition. Almost all hard constraints, when represented in the form of an integer linear programming model, show a one-to-one correspondence between assignments, suggesting that the process can be specialized. Ideally, a genetic algorithm which adheres to the constraints of a UCSP search space would out-perform an integer linear programming system due to its more efficient search heuristics.

Beyond exact methods, several heuristic methods have been proposed for UCSP, including genetic algorithms. Wang et al. employed a genetic algorithm which factored hard constraints into it loss function. They were able to achieve some success with their adaptive method, however they were not able to achieve 100% satisfaction of all of their rules, and their algorithm had to go through the extra work of eliminating solutions that violated hard constraints. Kumargazhanova et al. took a similar approach when solving a variation of UCSP concerning the breakdown of classes into smaller groups. Precise results were not given, however their method of breaking down UCSP and their loss functions were very similar to that of Wang et al. Both of these methods could be improved if their associated hard constraints did not need to be factored into their loss functions.

El-Sherbiny et al. took a different approach to address the problem of hard constraints. Their variation of the genetic algorithm circumvented the crossover operator, and relied only on a set of mutation operators. They took this approach since they reasoned that the crossover operator would not yield well-performing individuals. This drastic modification takes away the most distinguishing feature of the genetic algorithm, hampering its effectiveness. This approach is more akin to a particle swarm optimizer, however a more desirable method would include a recombination operator that does not generate poor solutions.

3. Methodology

This section quantifies the exact problem that is optimized by a genetic algorithm and explains the novel decoding method. The mechanics of random-keys are also clarified in this section. Afterwards, empirical results are presented, followed by a mathematical analysis of the method proposed in this section.

3.1. Problem Formation

The precise formulation of UCSP can vary from author to author, but this article only considers the following constraints:

1. One teacher per class, per room, per time slot.
2. A class's size cannot exceed room capacity.
3. Each class must meet a certain number of times per week.
4. Each class will meet in the same room during the same period of the day.

The following soft constraints are also considered:

1. Teacher preferences regarding the classes they teach.
2. Teacher time slot preferences.
3. Room utilization.

To calculate the first soft constraint, each teacher ranks the classes they would prefer to teach, with the first class being most desirable. If the class is not on the list, then it is assumed that it is not desired at all. Then, the classes a teacher is assigned are compared against their list, with the index of the class on their list (starting from zero) divided by the size of their list representing the loss value. If the class is not on the teacher's list, then the loss value is set to 1. The

loss values are then averaged for every class-teacher pairing. Unlike the class preferences, periods are not ranked. Thus, if a teacher is assigned a period not on their list, then the loss value is 1 else it is 0. These values are then averaged across all class-teacher pairings. Finally, room utilization is calculated with the following equation:

$$\text{loss} = \frac{\text{capacity} - \text{classSize}}{\text{capacity}},$$

and averaged across all class-room pairings.

3.2. Random-keys

The novel method relies on a specific decoder for random-keys. In all cases for this article, random-keys are used to define an assignment of items in a source list to items in a target list. This is accomplished by sorting the elements in the random-key, and tracking their original and final indices through the sort. The final indices are then scaled down to fit within the range of the target list if needed. The original index gives the item in the source list, and the final index gives the item in the target list. An example of this process will be given with the source and target lists (a, b, c, d, e) and $(1,2,3,4,5)$.

Index	0	1	2	3	4
Key	0.43	0.68	0.11	0.73	0.28
Sorted	0.11	0.28	0.43	0.68	0.73

With this, an index table can be generated, where i_s is the index in the source list and i_t is the index in the target list, and with that the resulting assignments can be listed.

Key Item	i_s	i_t	Assignment
0.11	0	2	$(a,3)$
0.28	1	4	$(b,5)$
0.43	2	0	$(c,1)$
0.68	3	1	$(d,2)$
0.73	4	3	$(e,4)$

A more complicated variation of this process can be applied to the situation where the target list is two-dimensional, i.e. a set of sets. In this case, the random-key reorders the elements in the source set, essentially mapping each element to a new index. The reordered source is then iterated over. For each element in the source set, the first viable pairing with a set within the target is formed, and then the assigned elements from the chosen set are removed. In this sense the random-key acts like a priority list or queue. It is critical that there are sufficient elements in the source set so that each element in the target has its number of potential assignments exhausted. If there are not sufficient elements in the source set, then "buffer" elements can be added which remove elements as if they have been assigned.

3.3. Novel Decoder

The initial step in the decoding process is to assign teachers to classes, which is simply done by using a random-key to map the array of classes onto the array of teachers, linearly scaling indices up or down so that each teacher is assigned and equivalent number of classes. After this pairing is completed, the more complex task of assigning rooms and time-slots is executed.

Before any assignment is done, the rooms are sorted by decreasing capacity, and each room is duplicated p times before all copies are inserted into a list, where p is the number of periods in a day. This list effectively contains pairings of rooms with periods, where the period can be derived from an index. Unless otherwise specified, the term "room" will refer to one of the rooms in the list going forward, meaning that it implicitly has a specific period assigned to it. The set of class-teacher pairings is then sorted be decreasing class size, with ties being broken by the number of times a class meets per week, again in decreasing order.

The one-to-one mapping of random-keys is exploited to satisfy the first hard constraint, and the data structure above satisfies the fourth, however the second and third constraints present more difficulty. The key observation of this article is that these constraints can be satisfied simply through the order in which class-teacher pairings are assigned to rooms and time slots. To demonstrate this, consider an example with two classrooms, one with a capacity of 50 and

another with a capacity of 100, and two classes, one with a size of 50 and another wise a size of 100. The class with 100 students can only occupy the second room if the capacity constraint is to be satisfied, however the smaller class can occupy either room. This presents a problem since if the smaller class is assigned to a room first, and it is assigned to the larger room, then the 100-student class has no feasible assignment even though it is evident that one exists. The solution to this is to assign the largest classes that meet the most often first, and the smallest, most infrequent classes last. Going back to the example, the 100-student class would have priority since its only option is the larger room, so the larger class would be assigned to that room. The 50-student class would then have a choice of both rooms, but since the larger room is already occupied, the smaller class would be assigned to the smaller room. Through this method, both the the second and third hard constraints are satisfied.

Returning to the general case, the lists are sorted and the smallest subset of compatible class-teacher pairings and rooms are selected. “Compatible” in this sense means that any non-overlapping assignment of the two sets would satisfy the second and third hard constraints. Since the divisions of the class-teacher pairings and the rooms remains consistent across iterations, a single random-key can be used. Since each room stores with it a list of the days of the week in which it is available, the rooms act as a two-dimensional target and the classes as a source. A section of the random-key is used to perform a priority mapping to create the assignment. This is repeated for each of the subset pairings. Once this is finished, the assignments are translated into a different data structure to represent the completed schedule.

The rest of the genetic algorithm is fairly standard. A population of 100 individuals was used, and the mutation rate was set to 5%. During an iteration of the algorithm, loss values for each individual in the population were calculated and used to carry out roulette wheel selection. Enough individuals were selected for reproduction to fill 60% of the resulting population. To produce offspring, the parents’ chromosomes were recombined with uniform crossover. 30% of the next generation was created by copying over the elites of the previous generation, and the remaining 10% was filled by randomly generating new individuals to keep diversity. The above method was tested on a relatively small data set from a math department at a local college-prep school, and the results are presented in the next section.

4. Experimental Results

The method above was run for 2,500 iterations 100 times and the results were averaged across all runs. Completing all 2,500 iterations in a single run took about 20 seconds real-time, however the code could be optimized further to cut back on overhead. The input data set was constructed to deliberately have a minimum loss value above zero. Although the exact value is not known, it is estimated to be around 0.5.

Figure 1: Loss vs. Iterations

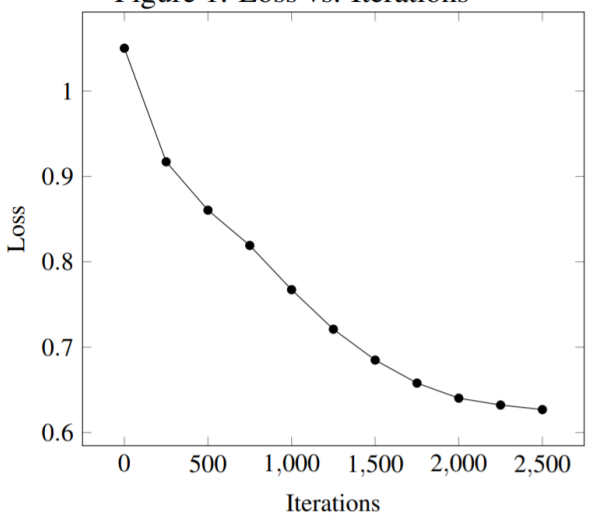

Figure 2: Parts vs. Iterations

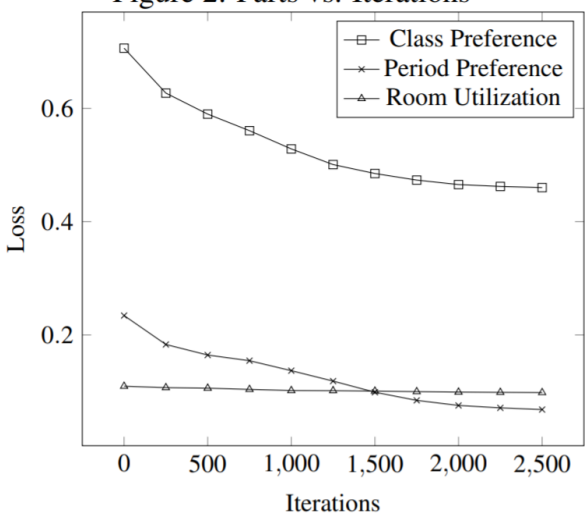

Figure 1 shows the decrease in the overall loss value of the most-fit individual. The biggest improvement is realized in the first 250 to 500 iterations, and then the improvement rate remains steady for another 1,000 iterations until it begins to level off. During the trials, it was noted that there were two local minima that the GA tended to settle on:

one with a loss value of 0.64 ± 0.03 and the other with a loss value of 0.57 ± 0.03. As demonstrated by the data, it was far more common for the algorithm to settle on the minimum with the higher loss value. Given more iterations, the minimum with the smaller loss value was often found.

Figure 2 shows the breakdown of the overall loss value into its component parts. As stated before, the composite loss value is generated from three smaller values: teachers' class preferences, teachers' teaching time preferences, and room utilization. Each of these factors was equally weighted when calculating the overall loss value, causing the component with the largest magnitude to have the largest effect on selection. The room utilization measurement had very little effect on the algorithm. Although it cannot be seen clearly on the graph, there was a slight decrease in the room utilization loss over the 2,500 iterations. The interplay of the equal weighting can also be seen in that the period preference loss decreases at a faster rate as the class preference loss levels out. The algorithm may be able to converge at a faster rate if different weightings of the components are employed.

5. Analysis

The key observation that this paper exploits is the unique properties of the random-key chromosome representation. This section explores these properties and gives a mathematical proposition as to why the proposed method is robust. A few notes on notation:

- Unless otherwise specified, all "sets" are assumed to be ordered sets. Ordered sets still cannot contain duplicate elements.
- $A \cup B$ denotes a union of two ordered sets where the resultant set can have any arbitrary ordering of its elements.
- Random-key chromosomes will be represented as an ordered set of reals rather than a vector.

It is assumed that random-keys are robust because they are unbiased in the representations that they encode. In other words, a decoder of a random key should not preferentially favor certain characteristics in the resulting output type. This is defined concretely in terms of uniform-assignment functions, and then this theory is applied to the proposed methodology to show that there is no bias.

5.1. Foundations

Definition 1: *Given an ordered set R, if $R \neq \emptyset$, and each element of R is an independent sampling of the uniform distribution of reals on the interval $[0,1)$, then R is a random-key set.*

Random-key sets are almost always paired with a mapping function or decoder. For the purposes of this article, a specific type of mapping function, called a uniform-assignment function, will be defined. Uniform-assignment functions group each element in a source set A with one or more elements in another source set B based on a random-key set R, such that each feasible grouping has an equal probability. The set of all feasible groupings is denoted by the set T. The set of all possible random-key sets for a given decoder is denoted as $\Re$, however it is possible for a uniform-assignment function to accept sets of random-key sets as long as the following definition is satisfied.

Definition 2: *Given a function $f: A \times \Re \to T$, f is a uniform-assignment function if given a random element $a \in A$ and a random element $t \in T$, $\forall R \in \Re, P(f(a,R) = t) = \frac{1}{|T|}$.*

One example of a uniform-assignment function is the simple direct mapping mentioned in the methodology section. The proof of this is as follows. Given two ordered sets, A and B, such that $|A| = n$, $|B| = m$, and $n \geq m > 0$, let $T = A \times B$ and $\Re = \{R \mid R \text{ is a random} - \text{key set and } |R| = n\}$. Fix $R \in \Re$, and choose a random element $a_1 \in A$ where a_1 is the i-th element of A, and a random pair $(a_2, b) \in T$ where b is the j-th element of B. Since $a_1 \in A$ and $a_2 \in A$,

$$P(a_1 = a_2) = \frac{1}{|A|} = \frac{1}{n}.$$

The probability that $f(a_2, R) = b$ is equivalent to the probability that the i-th element of R is greater than between $\frac{n(j-1)}{m}$ (inclusive) and $\frac{nj}{m}$ (exclusive) elements in R, which is equal to

$$\frac{1}{n}\cdot\left(\frac{nj}{m}-\frac{n(j-1)}{m}\right)=\frac{1}{m}.$$

Thus,

$$P\big(f(a_1,R)=(a_2,b)\big)=\frac{1}{n}\cdot\frac{1}{m}=\frac{1}{nm}=\frac{1}{|T|}.$$

This result will be slightly inaccurate if $\frac{n}{m}$ is not an integer due to rounding and/or flooring of fractional indices.

5.2. Extensions

Uniform assignment functions can be combined and keep their property of uniform assignment.

Theorem 1: *If* $g\colon A_1\times\Re_1\to T_1$ *and* $h\colon A_2\times\Re_2\to T_2$ *are both uniform-assignment functions, then the following is also a uniform-assignment function:*

$$\begin{aligned}f\colon\ &(A_1\cup A_2)\times(\Re_1\times\Re_2)\to T_1\cup T_2\\ &(a,R_1,R_2)\mapsto C(g(a,R_1),h(a,R_2))\text{ if }a\in A_1\cap A_2,\\ &(a,R_1,R_2)\mapsto g(a,R_1)\text{ if }a\in A_1,\\ &(a,R_1,R_2)\mapsto h(a,R_2)\text{ if }a\in A_2,\end{aligned}$$

where $C(x,y)$ *is a choosing function with equal probability of returning x or y.*

The proof of this is can be derived with a bit of work. Choose a random element $a\in A_1\cup A_2$, and a random element $t\in T_1\cup T_2$, and fix $(R_1,R_2)\in\Re_1\times\Re_2$.

$$\begin{aligned}P(f(a,R)=t)=&P(a\in(A_1\cap A_2)\cap C(g(a,R_1),h(a,R_2))=t)\\&+P(a\in A_1,a\notin(A_1\cap A_2)\cap g(a,R_1)=t)\\&+P(a\in A_2,a\notin(A_1\cap A_2)\cap h(a,R_2)=t).\end{aligned}$$

$$\begin{aligned}P(g(a,R_1)=t)&=P(t\in T_1)\cdot\frac{1}{|T_1|}\\&=\frac{|T_1|}{|T_1\cup T_2|}\cdot\frac{1}{|T_1|}\\&=\frac{1}{|T_1\cup T_2|}.\end{aligned}$$

$$\begin{aligned}P(C(g(a,R_1),h(a,R_2))=t)&=\frac{1}{2}\cdot\frac{|T_1|}{|T_1\cup T_2|}\cdot\frac{1}{|T_1|}+\frac{1}{2}\cdot\frac{|T_2|}{|T_1\cup T_2|}\cdot\frac{1}{|T_2|}\\&=\frac{1}{|T_1\cup T_2|}.\end{aligned}$$

$$P(a\in(A_1\cap A_2)\cap C(g(a,R_1),h(a,R_2))=t)=\frac{|A_1\cap A_2|}{|A_1\cup A_2||T_1\cup T_2|}.$$

$$\begin{aligned}P(a\in A_1,a\notin(A_1\cap A_2)\cap g(a,R_1)=t)&=\left(\frac{|A_1|}{|A_1\cup A_2|}-\frac{|A_1\cap A_2|}{|A_1\cup A_2|}\right)\cdot\frac{1}{|T_1\cup T_2|}\\&=\frac{|A_1|-|A_1\cap A_2|}{|A_1\cup A_2||T_1\cup T_2|}.\end{aligned}$$

$$P(a\in A_2,a\notin(A_1\cap A_2)\cap h(a,R_2)=t)=\frac{|A_2|-|A_1\cap A_2|}{|A_1\cup A_2||T_1\cup T_2|}.$$

$$\therefore P(f(a,R)=t)=\frac{|A_1|+|A_2|-|A_1\cap A_2|}{|A_1\cup A_2|\,|T_1\cup T_2|}$$
$$=\frac{1}{|T_1\cup T_2|}.\square$$

One can also ask whether a composition of uniform assignment functions yields another uniform assignment function, which as it turns out, it does.

Theorem 2: *If $f: A\times\Re_1\to B$ and $g: B\times\Re_2\to T$ are uniform assignment functions, then $g\circ f$ is also a uniform assignment function.*

Fix $a\in A$ and $t\in T$. The probability that $g(f(a))=t$ can be broken down into the probability that $f(a)=b$ and $g(b)=t$ for each $b\in B$. Thus the following constitutes the overall probability:

$$\sum_{b\in B}P(f(a)=b\cap g(b)=t)=\sum_{b\in B}\frac{1}{|B||T|}=\frac{1}{|T|}.$$

5.3. Nested Targets

A more complex proof is needed for the case of a function that maps a source set to subsets of sets within the target data set, $\mathfrak{D}$. One more small theorem is needed for this proof.

Theorem 3: *If R is a random-key set, $\forall L\subset R, L\neq\emptyset$, L is also a random-key set.*

Since R is a set of samplings from the uniform distribution of reals on the interval $[0,1)$, it is clear that any subset of R also meets this criterion.

The target data, $\mathfrak{D}$, is structured as sets within a set, with the inner sets potentially varying in size. Through the mapping process, elements in a source set, A, are paired with subsets of a fixed size, l, of the sets within the target data set. It is assumed that the following condition is held: $\forall D\in\mathfrak{D}, |D|\geq l$. The mapping procedure requires two random keys, one for selecting the set within the target data set, and one for selecting the subset of the chosen set. These will be denoted as $\Re_1$ and $\Re_2$ respectively. The keys have the following sizes:

$$\forall R_1\in\Re_1, |R_1|=|A|;$$
$$\forall R_2\in\Re_2, |R_2|=\max_{D\in\mathfrak{D}}|D|.$$

The size of the source set must also satisfy the following:

$$|A|=\sum_{D\in\mathfrak{D}}\left\lfloor\frac{|D|}{l}\right\rfloor.$$

If this condition is not satisfied, then buffer elements need to be added as mentioned in the methodology section.

The process for this type of assignment is quite different than the kinds previously discussed. The priority list, or queue process defined in the methodology section can be used for both assignments. Once an assignment to elements within a set in $\mathfrak{D}$ is completed, those elements cannot be assigned again. If fewer than l elements remain in a set, then it is skipped over.

If the target set is constructed from the global data set $\mathfrak{D}$, then it quickly becomes evident that the assignment function is not uniform, however in doing that, the wrong function is constructed. The assignment of elements in A is more analogous to a composition of functions, where the first function pairs elements in A with sets within $\mathfrak{D}$, and the second selects a subset of the set within $\mathfrak{D}$. Because of this, the target of the second function is dependent on the output of the first. These functions are defined as f and g respectively.

$$f: A\times\Re_1\to A\times\mathfrak{D}$$

Given $D\in\mathfrak{D}$,

$$g_D: A\times\Re_2\to A\times\{F\mid F\subset D \text{ and } |F|=l\}$$

The first fact that must be proven is that f is a uniform-assignment function. Because of the size of A in relation to $\mathfrak{D}$, the queue-like mapping is equivalent to a direct mapping. This is because by assigning every element in A to a subset of a set within $\mathfrak{D}$, every set within $\mathfrak{D}$ will have fewer than l unassigned elements remaining after every element in A is given an assignment. Hence, f is a uniform-assignment function.

For the proof that g is also a uniform-assignment function, g must first be constructed from the output of f. In a sense, the source of g is $(A \times \mathfrak{D}) \times \mathfrak{R}_2$, however the element from $\mathfrak{D}$ is more of a meta-parameter, thus it is denoted as a subscript and not included in the source set. The process by which g_D selects a subset from D is similar to the queue method. The elements in D are reordered according to a random-key $R \in \mathfrak{R}_2$, and the first l elements are used. Since the size of D can be smaller than the size of R, the key must be truncated. By theorem 3, the first $|D|$ elements of R can be used to form the random-key, and the function will retain its property of uniform-assignment. With this information, a proof can be constructed.

Let $\mathfrak{F} = \{F \mid F \subset D \text{ and } |F| = l\}$. Order is not accounted for when removing duplicate elements from $\mathfrak{F}$. This is because the sets within $\mathfrak{F}$ correspond to the list of days available that each room maintains. Days of the week have a natural order, so any subset can always be re-ordered in a consistent manner. Fix $(a_1, R_2) \in A \times \mathfrak{R}_2$ and fix $D \in \mathfrak{D}$ and $(a_2, F) \in \mathfrak{F}$. As demonstrated earlier, $P(a_1 = a_2) = \frac{1}{|A|}$. Since the reordering of D is uniform, the result of taking the first l elements of the reordered set is equivalent to taking l elements randomly from D without replacement. Because of this, the probability that the outputs match can be calculated as follows:

$$P(g_D(a_2, R_2) = F) = \binom{|D|}{|F|}^{-1} = \binom{|D|}{l}^{-1}.$$

Thus the overall probability, $\frac{1}{|A|}\binom{|D|}{l}^{-1}$, is equivalent to $\frac{1}{|A \times \mathfrak{F}|}$. Furthermore, the uniformity remains invariant across all values of the meta-parameter D.

Since f and g are uniform-assignment functions, and since g retains its property of uniform-assignment for all $D \in \mathfrak{D}$, the "composition" of these two functions is also a uniform-assignment function.

5.4. Application to Methodology

As demonstrated above, the initial mapping of classes to teachers is a uniform assignment. The more complicated mechanics take place after this point. The class-teacher pairings are mapped to rooms with implicit periods in sections, with the class-teacher subsets being disjoint. This process is not entirely clear-cut due to the mutability of the set of rooms. If the first subset of class-teacher pairings is assigned to a specific subset of rooms and periods, then that will affect the possible assignments for the following subsets of class-teacher pairings. Ultimately, however, regardless of which slots are used at which point in the process, the union of the target sets for every individual assignment function will yield the same, global target set. Thus, by theorem, the result of combining each separate assignment yields a uniform-assignment function overall. Lastly, by theorem, the composition of the class-teacher assignment with the room assignment yields a uniform-assignment function. This proves overall that the proposed methodology is unbiased in its conversion from random-keys to schedules.

6. Conclusion

This article presented a novel method of decoding random-keys to produce class schedules that satisfied a set of hard constraints. Empirical results were presented that demonstrated the viability of this method, and then mathematical analysis was employed to demonstrated that the method was unbiased. Overall this novel method appears to have promise going forward.

Future research should be done to find new search space restriction methods and other areas of application. The methods employed here are unfortunately limited to the scope of UCSP and related problems, however some of the basic concepts can be applied elsewhere. Since it was shown that uniform randomness, probabilistically speaking, can be preserved through a wide variety of transformations, more inventive decoders can be developed and tested. The one drawback with these search space restricting decoders is their computational load. They are far less efficient than index-related decoders, and thus take significantly longer to translate a representation into an actual data structure.

Methods of lowering the computational load of these decoders should also be researched if they are to be applied in time-sensitive scenarios.

References

Alvarez-Valdes, R., Crespo, E., & Tamarit, J. M. (2002). Design and implementation of a course scheduling system using Tabu Search. *European Journal of Operational Research*, 137(3), 512–523.

Bardadym, V. A. (1996). Computer-aided school and university timetabling: The new wave. In E. Burke & P. Ross (Eds.), Practice and theory of automated timetabling. *Lecture notes in computer science* (Vol. 1153, pp. 22–45). Berlin: Springer.

Bean, J. C. (1994). Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, 6(2), 154–160.

De Jong, K. A., & Spears, W. M. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1), 1–26.

El-Sherbiny, M., Zeineldin, R., & El-Dhshan, A. (2015). Genetic Algorithm for Solving Course Timetable Problems. *International Journal of Computer Applications*, 124(10), 1–7

Kumargazhanova, S., Suleimenova, L., Fedkin, Y., & Urkumbaeva, A. (2019, October). Development and Implementation of an Automation Algorithm for Class Scheduling Process at Universities. *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*

Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and Its Applications*, 391(6), 2193–2196.

Pishvaee, Mir Saman, and Mohamadreza Fazli Khalaf. "Novel robust fuzzy mathematical programming methods." *Applied Mathematical Modelling*, vol. 40, 2016, pp. 407–418.

Samiuddin, J., Haq, M.A. A novel two-stage optimization scheme for solving university class scheduling problem using binary integer linear programming. *Oper Manag Res 12*, 173–181 (2019).

Shiau, D.-F. (2011). A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences. *Expert Systems with Applications*, 38(1), 235–248.

Snyder, L. V., & Daskin, M. S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174(1), 38–53.

Wang, P., Xu, X., & Chuhong, L. An Improved Adaptive Genetic Algorithm and Its Application in Intelligent Course Scheduling System. *6th International Conference on Information Science and Control Engineering*, pp. 121–125.

Wang, Y. (2002). An application of genetic algorithm methods for teacher assignment problems. *Expert Systems with Applications*, 22(4), 295–302.

Biographies

Ian Smith is a senior at Cranbrook Kingswood Upper School, Bloomfield Hills, MI, USA. He is focusing on studying computer science and mathematics, and anticipates dual-majoring in computer science and pure math during in his undergraduate studies. He hopes to pursue a doctorate and teach computer science at a collegiate level. Outside of school, he enjoys employing these subjects in robotics and independent research projects. Recently, he started a chapter of the non-profit organization Codivate, a student-run organization which aims to introduce grade school students to computer programming, in the state of Michigan. He enjoys having the opportunity to teach in high school.

Dr. George Pappas is an Assistant Professor of Electrical and Computer Engineering at Lawrence Technological University, Southfield, MI, USA. He has also taught Biomedical Engineering courses in biomedical devices and imaging processing. He has over 10 years of teaching and research experience in embedded systems. He has been the PI for a recent DENSO grant in machine vision safety systems in vehicles. He has been with the Electrical and Computer Engineering Department since 2016. He received his masters and Ph.D. from Oakland University, Rochester MI, USA. He has taught and mentored students in the areas of embedded systems, encryption and security, imaging processing in medical and automotive application, microcontrollers, and High-Performance Computing systems, artificial intelligence and machine learning algorithms.