mathematics

MDPI

Article

Gene-Similarity Normalization in a Genetic Algorithm for the Maximum *k*-Coverage Problem

Yourim Yoon [1] and Yong-Hyuk Kim [2,*]

[1] Department of Computer Engineering, College of Information Technology, Gachon University, 1342 Seongnamdaero, Sujeong-gu, Seongnam-si, Gyeonggi-do 13120, Korea; yryoon@gachon.ac.kr

[2] School of Software, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea

* Correspondence: yhdfly@kw.ac.kr

Received: 28 February 2020; Accepted: 30 March 2020; Published: 2 April 2020

Abstract: The maximum *k*-coverage problem (MKCP) is a generalized covering problem which can be solved by genetic algorithms, but their operation is impeded by redundancy in the representation of solutions to MKCP. We introduce a normalization step for candidate solutions based on distance between genes which ensures that a standard crossover such as uniform and *n*-point crossovers produces a feasible solution and improves the solution quality. We present results from experiments in which this normalization was applied to a single crossover operation, and also results for example MKCPs.

Keywords: maximum *k*-coverage; redundant representation; normalization; genetic algorithm

1. Introduction

The maximum *k*-coverage problem (MKCP) is regarded as a generalization of several covering problems. The problem has a range of applications in combinatorial optimization such as scheduling, circuit layout design, packing, facility location, and covering graphs by subgraphs [1,2]. Recently, besides some theoretical approaches [3,4], it has been extended to many real-world applications such as blog-watch [5], seed selection in a Web crawler [6], map-reduce [7], influence maximization in social networks [8], recommendation in e-commerce [9], sensor deployment in wireless sensor networks [10], multi-depot train driver scheduling [11], cloud computing [12], and location problem [13].

Let $A = (a_{ij})$ be an $m \times n$ 0-1 matrix, and let w_i be a weight applied to each row of A. The objective of MKCP is to choose k columns so as to maximize the sum of the weights of the rows that contain '1' and are also located in one of these k columns.

This problem can be represented formally as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} w_i \cdot I\left(\sum_{j=1}^{n} a_{ij}x_j \geq 1\right) \\ \text{subject to} \quad & \sum_{j=1}^{n} x_j = k \\ & x_j \in \{0,1\}, \quad j = 1,2,\ldots,n, \end{aligned}$$

where $I(\cdot)$ is an indicator function ($I(false) = 0$ and $I(true) = 1$).

We are concerned with the case that $w_i = 1$ for all i, when we say that a row is covered by the selected k columns if it contains '1' which is also located in one of these columns. In this case, MKCP is to find k columns that cover as many rows as possible.

MKCP has many real-world applications. For example, MKCP can be applied to the following practical applications about museum touring [14]: Suppose that a museum can operate only k guided

tour programs of visiting a set of exhibits among n possible programs due to some constraints such as operating costs. If there are m visitors and it is possible to predict whether or not each visitor i is satisfied with the experience of each program j (meaning a_{ij}), the problem of choosing exactly k programs to satisfy as many visitors as possible is exactly formulated as MKCP. Among the visitors, if it is preferred to satisfy specific visitors who have more importance such as VIPs, the problem can be modeled more elaborately by giving different weight (w_i) to each visitor i. Similarly, MKCP can be applied to other practical applications such as public safety networks and systems [15,16]. For example, we can consider a disaster management system in which n agencies are involved and m positions for Unmanned Aerial Vehicles (UAVs) are available to enable the communication between the agencies. In this situation, it can be easily investigated whether or not each agency i is covered when a UAV is placed in each position j (meaning a_{ij}). If only k UAVs are available for resource management, the problem of choosing exactly k positions of UAVs to cover as many agencies in the disaster area as possible can also be formulated as MKCP.

The NP-hardness of MKCP can easily be deduced from the NP-hardness of the minimum set covering problem (MSCP) [17]. Many meta-heuristics have been applied to the MSCP such as tabu search [18], genetic algorithms [19,20], particle swarm optimization [21], ant colony optimization [22], etc., but MKCP has been scarcely addressed. Some naïve greedy heuristics [1,7,23–25] and an improved local search [26], which do not scale well to large datasets, have been studied. To the best of our knowledge, there is only one meta-heuristic [27] that uses particle swarm optimization, applied to MKCP, and genetic algorithms have not been applied. The present authors previously conducted an initial investigation of MKCP [28], which we now extend. Because MKCP selects a fixed number of columns, the representation of solutions is simpler than it is in other covering problems, and this favors the adoption of a genetic algorithm (GA). When we apply a GA to the MKCP, it also has an interesting inherent property that each gene of a chromosome is not just a number but actually a column of a matrix, which we analyze and relate to the characteristic of its solution space. We go on to present a problem-specific normalization for use in a GA which takes account of the feasibility of solutions and improves the solution quality.

The remaining parts are organized as follows. We analyze the solution space of MKCP, and the representation of solutions in Section 2. Based on this analysis, in Section 3, we propose a normalization method for producing feasible and improved solutions. In Section 4, we present experimental results to assess the effectiveness of our method. Finally, we draw conclusions in Section 5.

2. Representation and Space of Solution to MKCP

When we solve a problem with GAs, the representation of chromosomes is an important issue. Representation depends on the problem and reflects the properties of the problem. One representation of a solution to MKCP is a vector of k integers representing column indices; another is a binary vector of length n, in which each element indicates whether or not the corresponding column is selected. Thus, when $k = 2$ and $n = 4$, the integer representation $(1, 4)$ is equivalent to the binary vector representation $(1, 0, 0, 1)$. We focus on the integer vector representation.

Using the integer vector representation, the encodings $(1, 4)$ and $(4, 1)$ both represent the same solution. The coverage, which is the value of the objective function, is also independent of this ordering. Since for the same solution there may be more than one different encoding, encoding space (genotypes) is different from solution space (phenotypes). We can consider permutations and combinations as genotypes and phenotypes, respectively. In the integer vector representation, it is natural to encode combinations using permutation encoding. However, in this case, many encodings correspond to the same combination. That is, this representation is redundant. Let G be the space of encodings in the integer vector representation, and let two encodings x and y in G be $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$, respectively. Now, the relation $\sim$ is defined on G:

Definition 1. *$x \sim y$ if and only if we have a permutation $\sigma \in \Sigma_k$ such that $\sigma(x_1, x_2, \ldots, x_k) = (y_1, y_2, \ldots, y_k)$, where Σ_k is the set of permutations of size k, and $\sigma(x)$ represents x permuted by σ.*

Proposition 1. *The relation $\sim$ becomes an equivalence relation.*

Proof. For each $i \leq k$, $\langle \Sigma_i, \circ \rangle$ forms a symmetric group S_i, where $\circ$ is the function composition operator [29]. Then, the direct product $P := \Pi_{i=1}^{k} S_k$ is also a group [29], and therefore has an identity meaning that the relation $\sim$ is reflexive. When $\sigma \in P$, its inverse $\sigma^{-1} \in P$ exists, i.e., the relation $\sim$ is symmetric. The group P is closed under the operator $\circ$: It means if $\sigma^1, \sigma^2 \in P$, then $\sigma^1 \circ \sigma^2 \in P$. That is, the relation $\sim$ is also transitive. Taken together, the relation $\sim$ becomes an equivalence relation. □

For example, the solutions $(1,3,5,7)$ and $(3,7,5,1)$ are equivalent. The number of vectors equivalent to a vector $g \in G$ is $k!$.

The equivalence relation of Definition 1 allows us to consider the real solution space (phenotype space) as the set of equivalence classes of elements in G, i.e., the quotient space $G/\sim$.

We can measure the similarity of two vectors x and y in G, the space of MKCP solution encodings, using a distance metric D, which we obtain by summing values of a subsidiary metric d which measures the difference between two columns of the matrix A which expresses the MKCP, as follows:

$$D(x,y) := \sum_{i=1}^{n} d(x_i, y_i). \tag{1}$$

Here, the genes of two chromosomes, x_i and y_i, represent chosen column indices. If we regard the indices as just labels, not column vectors, the discrete metric, which becomes one if the two indices are the same, and zero otherwise, can be used as a metric d. This satisfies all the conditions for a distance metric including the triangular inequality.

An alternative metric is Hamming distance, which is a measure of the dissimilarity of two binary vectors. If we consider each column of the matrix A as a binary vector (the column vector), not just an integer, we can find the Hamming distance between any two column vectors, and hence between the corresponding genes x_i and y_i in two chromosomes from G. These distances can then be summed as shown in Equation (1).

Figure 1 shows two distances in G calculated using discrete metric and Hamming distance. As shown in Figure 1c, the discrete metric simply compares the column indices of x_i and y_i, and ignores the contents of the corresponding columns of A. In Figure 1d, we see that the distance between x_1 and y_1 is the Hamming distance between the first and the second column vectors of A.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

(**a**) The 5×4 matrix A

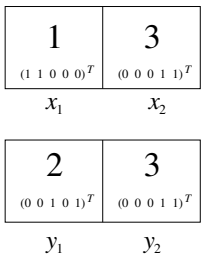
Gene 1 = $(1, 1, 0, 0, 0)^T$

Gene 2 = $(0, 0, 1, 0, 1)^T$

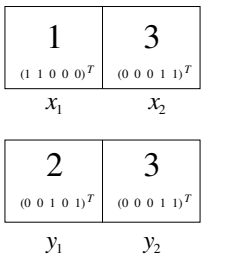
Gene 3 = $(0, 0, 0, 1, 1)^T$

Gene 4 = $(0, 1, 1, 0, 0)^T$

(**b**) Four column vectors (genes)



(**c**) Discrete metrics



(**d**) Hamming distances

Figure 1. Discrete metric vs. Hamming distance, for the chromosomes (1,3) and (2,3).

Now, we establish a metric in the quotient space $G/\sim$ by the following proposition:

Proposition 2. *Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ be in G, and let a metric D on G be defined by Equation (1). Then,*

$$\bar{D}(\bar{x}, \bar{y}) := \min_{\sigma \in \Sigma_k} \sum_{i=1}^{k} d(x_i, \sigma_i(y)) \tag{2}$$

becomes a metric on $G/\sim$, where $\sigma_i(y)$ represents the ith element of permuted y by σ.

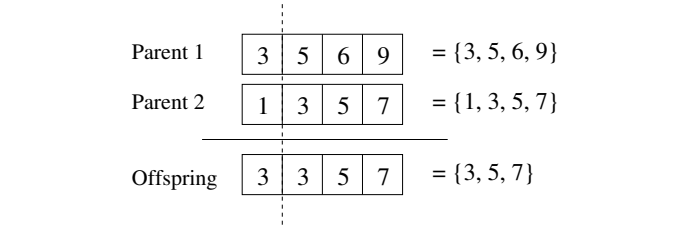
Proof. Let σ be in the group $\Pi_{i=1}^{k} S_k$. The computation $D(x, y) := \sum_{i=1}^{n} d(x_i, y_i)$, is unaffected by summation order, and thus $D(x, y) = D(\sigma(x), \sigma(y))$. Hence, σ is an isometry on G, and thus $\Pi_{i=1}^{k} S_k$ is an isometry subgroup. The relation $\sim$ is an equivalent relation obtained from $\Pi_{i=1}^{k} S_k$. Hence from [30,31] $\bar{D}(\bar{x}, \bar{y})$ is a metric on $G/\sim$. □

3. Normalization in MKCP

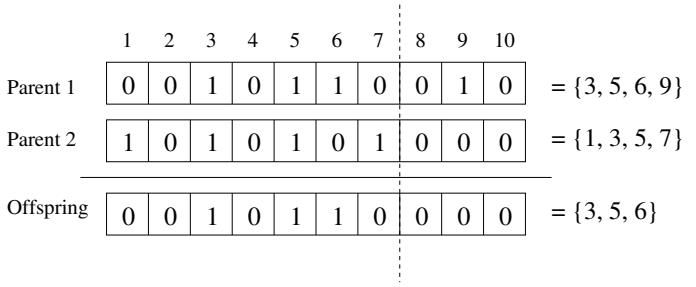
As shown above, redundancy in the integer representation of solutions to MKCP means that the encoding (genotype) space G is unnecessarily larger than the true solution (phenotype) space, which is the quotient space $G/\sim$. Redundant representations can be expected to reduce the performance of genetic algorithms significantly, which, in particular, undermines the effectiveness of standard crossovers defined using masks [32]. The problem of redundant representations have been addressed by a number of methods such as adaptive crossover [33–36], and among which the normalization technique [37] is representative. Normalization changes a parent genotype into a different genotype with the same phenotype which is similar to the genotype of the other parent before a standard crossover is performed. It is based on adaptive crossovers [34,35] and many variants have appeared [31,38,39].

3.1. Preserving Feasibility

A representation is infeasible if it does not meet the requirements of a solution. Figure 2a shows an integer representation which is infeasible because it contains duplicate column indices, and Figure 2b shows a binary representation which is infeasible because it contains the wrong number of '1's. Figure 2 also shows how these infeasible solutions are likely to be created by standard crossover operators such as uniform and n-point crossovers.



(**a**) Integer encoding
($n = 10$ and $k = 4$)



(**b**) Binary encoding
($n = 10$ and $k = 4$)

Figure 2. How infeasible solutions can be created by crossover operations.

A repairing step can be performed to restore feasibility, but this has the effect of a mutation, and may garble gene sequences inherited from parents. Using integer encoding, the problem can be avoided by rearranging the parents so that any shared column indices are in the same position. Such rearrangement naturally makes offspring preserve feasibility and no special repairing step is required. This form of normalization is easily implemented, as shown in the pseudocode in Figure 3, and takes $O(k^2)$ time.

```
FP_normalization(x, y)
{
    for i ← 0 to k
        for j ← 0 to k
            if x_i = y_j, then
                Swap values of y_i and y_j;
}
```

Figure 3. Pseudocode of normalization to preserve feasibility.

In Figure 4, this normalization to preserve feasibility for the example in Figure 2a is shown.

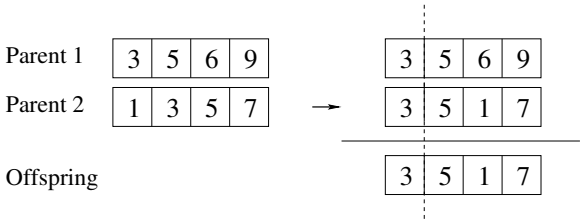


Figure 4. Normalization to preserve feasibility in the example from Figure 2a.

3.2. Normalization for Improving Solution Quality

A good solution to MKCP will consist of dissimilar columns. For example, in Figure 1, $(1,3)$ is a better solution than $(1,4)$ because the '1's in Columns 1 and 3 all occur in different rows, while Columns 1 and 4 share a '1' in Row 2.

In the context of a genetic algorithm, we would expect chromosomes with genes corresponding to dissimilar columns to be most effective. Looking again at Figure 1, suppose that we apply a standard one-point crossover to the parents $(1,2)$ and $(3,4)$. If the cutting line lies between the first gene and the second one, then the offspring will be $(1,4)$, and this offspring covers the rows $\{1,2,3\}$ of A. However, if the second parent is rearranged to $(4,3)$, the offspring becomes $(1,3)$, which covers $\{1,2,4,5\}$. This example is illustrated in Figure 5. In general, we want the offspring to have genes that correspond to columns that are as dissimilar as possible. Thus, it is helpful to rearrange chromosomes so that genes corresponding to similar columns are in the same positions. The goal of this rearrangement can be formulated in terms of distance in the phenotype space $G/\sim$. Consider two parents x and y in the genotype space G. Rearranging genes so that those corresponding to the most similar columns are located in the same positions is equivalent to the search for a permutation σ^* such that the distance between the equivalence class of x and that of y is equal to the distance between x and $\sigma^*(y)$. This is also equivalent to finding the σ which minimizes the distance between x and the permuted y in Equation (2).

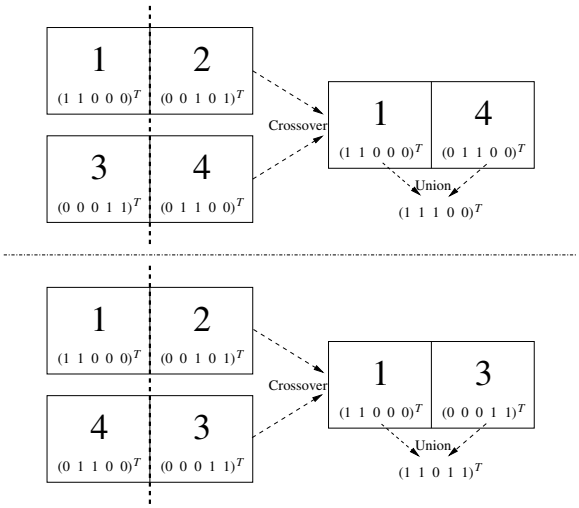


Figure 5. Effect of rearranging genes to match similar columns in the example from Figure 1.

The optimal rearrangement is achieved by considering all of the permutations of genes in the second parent, and choosing the permutation that minimizes the distance sum between the column vectors corresponding to gene pairs in the same locations. If Hamming distance H is used to get the dissimilarity between the column vectors corresponding to two genes, we will choose the permutation σ^* such that

$$\sigma^* = \underset{\sigma \in \Sigma_k}{\operatorname{argmin}} \sum_{i=1}^{k} H(x_i, \sigma_i(y)), \tag{3}$$

where Σ_k is the set of all the permutations of length k and $\sigma_i(y)$ denotes the ith element of permuted y by σ.

We give an example case in Figure 6, in which the chromosomes $(1, 2)$ and $(3, 4)$ are both parents, and now we normalize $(3, 4)$. Because k is 2, the number of all the permutations is just 2. We compute $\sum_{i=1}^{k} H(x_i, \sigma_i(y))$ for each permutation. If $\sigma^1 = \binom{1\ 2}{1\ 2}$, $\sigma^1(y) = (3, 4)$. Then, $H(x_1, \sigma_1^1(y)) = 4$, $H(x_2, \sigma_2^1(y)) = 2$, and their sum is 6. For the second permutation $\sigma^2 = \binom{1\ 2}{2\ 1}$, $\sigma^2(y) = (4, 3)$. Then, $H(x_1, \sigma_1^2(y)) = 2$, $H(x_2, \sigma_2^2(y)) = 2$, and their sum is 4. Since this is smaller than 6, $\sigma^2 = \binom{1\ 2}{2\ 1}$ is the optimal permutation.

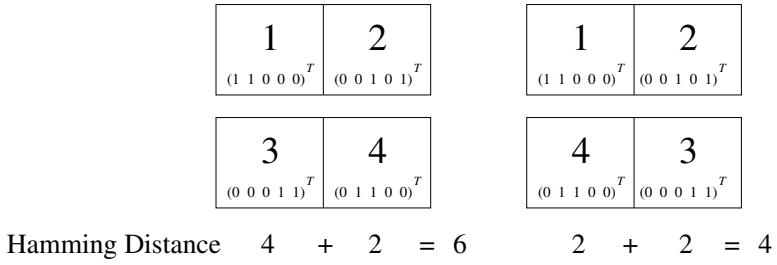


Figure 6. Normalization with Hamming distance in the example from Figure 1.

Enumerating all $k!$ permutations is intractable for a large k, and thus this procedure is intractable. However, the problem can be solved using the Hungarian method [40], which is a network-flow-based technique. It provides an optimal result, and runs in $O(k^3)$ time [41]. Alternatively, we can use a fast heuristic [42], which runs in $O(k^2)$ and produces results very close to the optimum. Either of these methods can be treated as a function that accepts a 2D array, in which the elements are the distances between two genes, and returns the permutation with the minimum total distance between chromosomes. This normalization is shown in the pseudocode of Figure 7.

```
OPT_normalization(x, y)
{
    for i ← 0 to k
        for j ← 0 to k
            D[x_i, y_j] ← H(x_i, y_j);
    σ* ← argmin_{σ∈Σ_k} Σ_{i=1}^{k} H(x_i, σ_i(y)); // using the Hungarian method or its fast variant
    y ← σ*(y);
}
```

Figure 7. Pseudocode of normalization by the optimal rearrangement.

After we rearrange the second parent according to the permutation σ^*, a standard crossover is applied.

Now, we investigate the relation between this optimal rearrangement and the feasibility. There may be more than one optimal rearrangement satisfying Equation (3), but we can show that one of them is always feasible. Feasibility is preserved by an optimal permutation σ^* which locates all common indices in the same positions by σ^*, as we now prove:

Proposition 3. *If $x = (x_1, x_2, \ldots, x_k)$ and $y = (y_1, y_2, \ldots, y_k)$ are two chromosomes and $x_p = y_q$, then there exists $\sigma^* \in \Sigma_k$ such that $\sigma^*_p(y) = y_q$ and $\sum_{i=1}^{k} d(x_i, \sigma^*_i(y)) \leq \sum_{i=1}^{k} d(x_i, \sigma_i(y))$ for all $\sigma \in \Sigma_k$, where d is a distance metric.*

Proof. Let σ' be $\text{argmin}_{\sigma \in \Sigma k} \sum_{i=1}^{k} d(x_i, \sigma_i(y))$. There is an index r satisfying $\sigma'_r(y) = y_q$. Let σ'' be the same permutation as σ', except that $\sigma'(y)_p$ and $\sigma'_r(y)$ are exchanged:

$$\sigma''_i(y) = \begin{cases} \sigma'_r(y) & \text{if } i = p, \\ \sigma'_p(y) & \text{if } i = r, \\ \sigma'_i(y) & \text{otherwise.} \end{cases}$$

Then,

$$\begin{aligned}
\sum_{i=1}^{k} d(x_i, \sigma''_i(y)) &= d(x_p, \sigma''_p(y)) + d(x_r, \sigma''_r(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma''_i(y)) \\
&= d(x_p, \sigma'_r(y)) + d(x_r, \sigma'_p(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma'_i(y)) \\
&= d(x_p, y_q) + d(x_r, \sigma'_p(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma'_i(y)) \\
&= d(x_r, \sigma'_p(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma'_i(y)) \quad (\because d(x_p, y_q) = 0 \text{ by assumption}) \\
&\leq d(x_r, x_p) + d(x_p, \sigma'_p(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma'_i(y)) \quad (\because \text{triangular inequality}) \\
&= d(x_r, y_q) + d(x_p, \sigma'_p(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma'_i(y)) \\
&= d(x_r, \sigma'_r(y)) + d(x_p, \sigma'_p(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma'_i(y)) \\
&= \sum_{i=1}^{k} d(x_i, \sigma'_i(y)) \\
&\leq \sum_{i=1}^{k} d(x_i, \sigma_i(y)) \text{ for all } \sigma \in \Sigma_k.
\end{aligned}$$

Hence, $\sum_{i=1}^{k} d(x_i, \sigma_i''(y)) \leq \sum_{i=1}^{k} d(x_i, \sigma_i(y))$ for all $\sigma \in \Sigma_k$ and $\sigma_p''(y) = y_q$. □

This proof relies on the triangular inequality, which is a key property of any valid distance. Thus, Proposition 3 holds for distances other than Hamming distance. We could use discrete metric, introduced in Section 2. This distance provides a very rough comparison of two solutions, but Proposition 3 still holds. Equation (3) can be rewritten using discrete metric ρ instead of Hamming distance H:

$$\sigma^* = \underset{\sigma \in \Sigma_k}{\operatorname{argmin}} \sum_{i=1}^{k} \rho(x_i, \sigma_i(y)). \tag{4}$$

In particular, in this case, feasibility is preserved by any optimal permutation σ^* in Equation (4). Its proof is quite similar to the proof of Proposition 3.

Proposition 4. *If $x = (x_1, x_2, \ldots, x_k)$ and $y = (y_1, y_2, \ldots, y_k)$ are two chromosomes and $x_p = y_q$, then $\sigma_p^*(y) = y_q$, where σ^* is a permutation such that $\sum_{i=1}^{k} \rho(x_i, \sigma_i^*(y)) \leq \sum_{i=1}^{k} \rho(x_i, \sigma_i(y))$ for all $\sigma \in \Sigma_k$.*

Proof. We assume that $\sigma_p^*(y) \neq y_q$. There is an index r satisfying $\sigma_r^*(y) = y_q$. Let σ' be the same permutation as σ^*, except that $\sigma^*(y)_p$ and $\sigma_r^*(y)$ are exchanged:

$$\sigma_i'(y) = \begin{cases} \sigma_r^*(y) & \text{if } i = p, \\ \sigma_p^*(y) & \text{if } i = r, \\ \sigma_i^*(y) & \text{otherwise.} \end{cases}$$

Then,

$$\begin{aligned}
\sum_{i=1}^{k} \rho(x_i, \sigma_i'(y)) &= \rho(x_p, \sigma_p'(y)) + \rho(x_r, \sigma_r'(y)) + \sum_{i \neq p, i \neq r} \rho(x_i, \sigma_i'(y)) \\
&= \rho(x_p, \sigma_r^*(y)) + \rho(x_r, \sigma_p^*(y)) + \sum_{i \neq p, i \neq r} \rho(x_i, \sigma_i^*(y)) \\
&= \rho(x_p, y_q) + \rho(x_r, \sigma_p^*(y)) + \sum_{i \neq p, i \neq r} \rho(x_i, \sigma_i^*(y)) \\
&= \rho(x_r, \sigma_p^*(y)) + \sum_{i \neq p, i \neq r} d(x_i, \sigma_i^*(y)) \quad (\because x_p = y_q \text{ by assumption}) \\
&< 1 + 1 + \sum_{i \neq p, i \neq r} d(x_i, \sigma_i^*(y)) \\
&= \rho(x_p, \sigma_p^*(y)) + \rho(x_r, x_p) + \sum_{i \neq p, i \neq r} \rho(x_i, \sigma_i^*(y)) \quad (\because x_p \neq \sigma_p^*(y) \text{ and } x_r \neq x_p) \\
&= \rho(x_p, \sigma_p^*(y)) + \rho(x_r, y_q) + \sum_{i \neq p, i \neq r} \rho(x_i, \sigma_i^*(y)) \\
&= \rho(x_p, \sigma_p^*(y)) + \rho(x_r, \sigma_r^*(y)) + \sum_{i \neq p, i \neq r} \rho(x_i, \sigma_i^*(y)) \\
&= \sum_{i=1}^{k} \rho(x_i, \sigma_i^*(y)).
\end{aligned}$$

This contradicts the assumption that $\sum_{i=1}^{k} \rho(x_i, \sigma_i^*(y)) \leq \sum_{i=1}^{k} \rho(x_i, \sigma_i(y))$ for all $\sigma \in \Sigma_k$. □

Using discrete metric will only cause identical indices to be rearranged into the same positions. In fact, normalization by discrete metric is exactly the same as rearranging for preserving feasibility introduced in Section 3.1.

4. Experiments

4.1. Test Sets and Test Environments

Our experiments were conducted on 65 instances of 11 set cover problems with various size and densities, from the OR-library [43]. Although these benchmark data were designed as set cover problems, the data can also be considered as maximum k-coverage problems, as in [27]. Some details of these problems are presented in Table 1, where m and n are the numbers of rows and columns, respectively, and density is the percentage of '1's in the MKCP matrix A. The present authors previously experimented with problems with fixed values of k of 10 and 20 [28]. However, in this study, we varied k with the tightness ratio α, which is the product of k and the density of a problem. The higher is the tightness ratio, the larger is the value of the object function, which is the coverage, that we are likely to achieve. If the tightness ratio is 1, the optimum coverage is likely to be very close to n. We used tightness ratios of 0.8, 0.6, and 0.4.

We implemented all our tested algorithms in C language using *gcc* version 5.4.0, and ran them on Ubuntu 16.04.6.

Table 1. Problem set.

Problem Set	m	n	Density	#Instances Tested
I-4	200	1000	2%	10
I-5	200	2000	2%	10
I-6	200	1000	5%	5
I-A	300	3000	2%	5
I-B	300	3000	5%	5
I-C	400	4000	2%	5
I-D	400	4000	5%	5
I-E	500	5000	10%	5
I-F	500	5000	20%	5
I-G	1000	10,000	2%	5
I-H	1000	10,000	5%	5

4.2. Effect of Normalization on a Crossover

To see whether or not normalization is effective at crossover, we performed experiments with three methods of rearranging the second of two parents before a crossover:

- REPAIR: The second parent is not rearranged before the crossover, but infeasible offspring are repaired to restore feasibility after the crossover.
- FP: The second parent is rearranged to produce only feasible offspring using the normalization in Figure 3.
- OPT: The second parent is rearranged by the normalization in Figure 7, to minimize the sum of the distances using the permutation σ^* of Equation (3).

REPAIR produces feasible offspring by the method of replacing duplicate column indices with a randomly chosen one among the indices that are not contained in the offspring. OPT was implemented using the Hungarian method [40], which runs in $O(k^3)$ time.

To determine the most effective method, we performed the following steps:

1. N parent chromosomes were randomly generated. (N was set to 100.)
2. The chromosomes were randomly paired, making $N/2$ couples.
3. The second parent in each pair was rearranged using the methods described above.
4. A uniform crossover was applied to each couple.
5. We computed the mean and the standard deviation for the coverage of each of the $N/2$ offspring.

This procedure was applied to a single instance of each problem listed in Table 1 using REPAIR, FP, and OPT. Table 2 shows the results for each of these 11 instances. We see that OPT normalization outperforms the others, and can therefore be expected to improve the performance of a GA. Moreover, the results of the one-tailed *t*-test show that the objective values of offspring produced using OPT normalization were better than those of their parents. On the contrary, the values produced using REPAIR and FP were similar to those of their parents. This suggests that, compared to other methods, OPT normalization would strongly support the function of the crossover operator in searching the solution space in a promising direction, without replacement strategy.

Table 2. Effect of normalization on a single crossover, after the REPAIR, FP, and OPT procedures.

Tightness			Parents		REPAIR		FP		OPT	
Ratio	Instance	*k*	Ave	SD	Ave	SD	Ave	SD	Ave	SD
0.8	scp41	40	110.27	6.33	110.30	5.89	110.00	5.46	**113.80**	6.12
	scp51	40	110.72	6.62	110.00	6.95	111.82	6.90	**115.02**	6.77
	scp61	16	111.33	6.43	111.24	6.64	111.20	6.73	**114.20**	6.24
	scpa1	40	167.69	7.76	166.80	7.83	167.68	8.47	**173.66**	8.93
	scpb1	16	168.31	9.27	170.80	9.79	168.90	9.11	**171.36**	9.25
	scpc1	40	220.40	10.53	221.04	10.83	220.42	9.76	**225.84**	9.91
	scpd1	16	223.74	9.22	222.34	10.15	224.36	8.69	**228.56**	9.48
	scpnre1	8	284.43	10.23	285.08	10.93	285.76	11.08	**287.14**	9.27
	scpnrf1	4	295.44	10.76	294.94	9.94	294.44	9.50	**297.34**	9.75
	scpnrg1	40	553.30	14.49	553.96	14.56	553.82	16.10	**561.28**	12.16
	scpnrh1	16	560.94	14.49	562.12	16.57	563.14	16.24	**567.46**	17.09
0.6	scp41	30	90.40	6.41	90.80	6.34	89.98	4.82	**95.34**	6.14
	scp51	30	90.77	6.74	91.92	6.92	91.44	6.50	**94.20**	6.64
	scp61	12	90.96	6.54	90.56	6.17	90.94	5.86	**91.88**	7.75
	scpa1	30	137.54	8.09	136.74	8.95	137.60	7.63	**143.74**	8.01
	scpb1	12	138.05	8.31	138.44	8.36	139.68	9.08	**141.68**	8.28
	scpc1	30	181.59	10.64	178.52	10.55	180.24	11.30	**186.06**	10.09
	scpd1	12	185.27	9.08	184.80	9.06	185.34	9.01	**188.30**	7.72
	scpnre1	6	233.02	11.34	231.60	12.83	234.04	12.41	**234.12**	11.04
	scpnrf1	3	244.40	11.37	245.04	10.48	244.24	10.28	**245.68**	9.57
	scpnrg1	30	452.91	14.48	452.70	14.52	452.58	14.07	**461.34**	13.42
	scpnrh1	12	461.87	13.66	460.80	16.68	462.22	16.58	**462.40**	15.61
0.4	scp41	20	66.44	5.52	66.84	5.59	67.30	6.09	**68.06**	5.19
	scp51	20	66.41	5.54	67.44	6.61	66.84	6.44	**68.98**	6.71
	scp61	8	66.55	6.80	67.68	6.05	66.72	5.91	**67.74**	5.71
	scpa1	20	100.82	8.40	99.80	7.62	100.00	7.63	**103.54**	6.80
	scpb1	8	101.25	7.43	101.74	8.91	101.06	7.38	**104.06**	8.43
	scpc1	20	132.47	10.42	131.68	8.78	133.14	9.07	**136.14**	9.92
	scpd1	8	135.48	10.49	136.32	8.49	134.72	9.46	**136.94**	9.96
	scpnre1	4	171.43	10.32	172.04	10.14	172.94	12.05	**173.16**	11.10
	scpnrf1	2	179.95	10.83	180.04	9.85	179.94	11.46	**182.90**	11.22
	scpnrg1	20	331.51	13.57	332.10	13.64	330.52	14.89	**337.30**	14.60
	scpnrh1	8	338.58	14.59	336.84	14.80	337.30	14.42	**343.44**	13.90
t-test *p*-value *			-		4.25×10^{-1}		1.09×10^{-1}		7.52×10^{-12}	

Ave and SD are the average and the standard deviation of the fitness of 100 parents (in the column of "Parents") or 50 offspring (in the remaining columns of REPAIR, FP, and OPT), respectively. REPAIR produces feasible offspring by random repair. FP rearranges the second parent to produce feasible offspring using the normalization in Figure 3. OPT is optimized normalization of the second parent. * The one-tailed *t*-test of the null hypothesis that the result of given method is equal to fitness of parents.

4.3. Performance of GAs with Normalization Methods

Our underlying evolutionary model is similar to the model of CHC [44], which was applied to many problems [45–50]. We paired a population of *N* chromosomes randomly, and then we applied

crossover to each pair, generating a total of $N/2$ offspring. We ranked all parents and offspring and the fittest N individuals among them became the population in the next generation. We used 100 as the size of population in our experiments. We reinitialized the population, except for the best individual, if there were no changes over $kr(1-r)$ generations, where r is a divergence ratio that wes set to 0.25. This GA stopped after 500 generations and returned the best it has found. The pseudocode of our GA is given in Figure 8.

```
Initialize a population P of N individuals;
for i ← to maximum generations
{
    Randomly pair a population P of N individuals;   {N/2 pairs}
    for each pair (p1, p2) ∈ P   {N/2 iterations}
    {
        Normalize p2 to make close to p1;   {optionally applied}
        o ← crossover(p1, p2);   {make offspring from parents}
    }
    P ← the best N individuals among N parents and N/2 offspring;
    if there are no changes in P over kr(1 − r) generations, then
        Reinitialize a population P, except for the best individual;
}
return the best individual;
```

Figure 8. Pseudocode of our genetic algorithm.

In the following experiments, we changed the normalization method in a single GA. We compared the output of our GA with the best result that we found in this study, using the metric %-gap, which is $100 \times |best - output|/best$.

Thirty trials were performed for each method, and the averaged results are shown in Table 3. The results labeled RR-GA were produced without normalization: infeasible offspring produced were repaired randomly using the same method as REPAIR in Section 4.2. We also compared the results from the GA with a multi-start method using randomly generated solutions. In each run of this method, called Multi-Start, we sampled 10^6 random solutions and chose the best one. Even RR-GA performed significantly better than Multi-Start, suggesting that GAs are an appropriate mechanism for solving the MKCP.

The results labeled FP-GA in Table 3 were produced by rearranging the genes of the second parent to produce feasible offspring without the need for repair. FP-GA outperformed RR-GA for large values of k but not for small values. It seems that the effect of mutation by repair is rather effective when the solution space is small.

The results labeled OPT-GA in Table 3 were produced by the GA with the proposed normalization. Using the same GA as FP-GA, OPT-GA rearranges the genes of the second parent to minimize the sum of distances between genes (column vectors) before applying recombination. OPT-GA clearly outperforms Multi-Start and RR-GA; the results of one-tailed t-tests prove that OPT-GA also outperforms FP-GA significantly.

The results of the t-tests show that Multi-Start is clearly the worst technique, even though it is allowed 10^6 evaluations, while the GA-based methods produce relatively small 2.5×10^4 chromosomes. RR-GA and FP-GA have similar performance, and OPT-GA clearly does best.

Table 3. Results of three GAs compared with random solutions to MKCP.

Tightness	Instance		Multi-Start		RR-GA		t-Test [1]	FP-GA		t-Test [2]	OPT-GA		t-Test [3]
Ratio	Set	k	%-Gap	Ave	%-Gap	Ave	p-Value	%-Gap	Ave	p-Value	%-Gap	Ave	p-Value
	I-4	40	28.17	141.35	7.08	182.86	6.62×10^{-16}	3.36	190.19	3.45×10^{-11}	**1.96**	192.95	2.56×10^{-10}
	I-5	40	28.67	141.52	6.84	184.82	2.92×10^{-19}	3.95	190.57	1.71×10^{-9}	**1.64**	195.14	5.16×10^{-10}
	I-6	16	20.56	143.77	4.22	173.35	8.21×10^{-10}	4.21	173.37	4.67×10^{-1}	**2.53**	176.42	6.86×10^{-5}
	I-A	40	27.62	207.57	6.16	269.13	1.09×10^{-9}	4.29	274.49	4.85×10^{-6}	**1.81**	281.61	5.45×10^{-6}
	I-B	16	20.56	208.93	4.52	251.10	2.69×10^{-7}	4.73	250.54	8.86×10^{-2}	**2.68**	255.94	1.05×10^{-4}
0.8	I-C	40	26.81	269.33	6.12	345.48	2.89×10^{-8}	4.85	350.16	7.50×10^{-5}	**2.10**	360.28	2.11×10^{-5}
	I-D	16	18.78	271.43	3.91	321.14	8.71×10^{-8}	4.15	320.31	6.58×10^{-2}	**1.90**	327.84	2.65×10^{-6}
	I-E	8	12.65	336.81	3.63	371.60	7.01×10^{-8}	4.04	369.99	9.37×10^{-3}	**2.65**	375.37	1.31×10^{-4}
	I-F	4	6.23	346.58	2.25	361.27	2.18×10^{-6}	2.42	360.66	1.27×10^{-1}	**1.83**	362.82	1.78×10^{-3}
	I-G	40	21.61	630.13	4.73	765.78	3.64×10^{-10}	3.82	773.12	4.74×10^{-4}	**1.64**	790.58	1.47×10^{-5}
	I-H	16	14.79	634.61	3.31	720.14	1.81×10^{-8}	3.75	716.83	1.29×10^{-2}	**1.93**	730.41	5.89×10^{-5}
	I-4	30	31.62	121.57	6.72	165.85	3.47×10^{-16}	4.75	169.35	8.86×10^{-9}	**2.69**	173.01	1.76×10^{-9}
	I-5	30	32.71	121.79	6.76	168.76	1.25×10^{-17}	5.69	170.70	1.25×10^{-5}	**2.53**	176.42	1.53×10^{-12}
	I-6	12	21.51	124.00	4.35	151.12	3.29×10^{-7}	4.21	151.35	1.62×10^{-1}	**2.56**	153.95	1.02×10^{-3}
	I-A	30	31.03	178.49	6.12	242.97	2.64×10^{-8}	5.30	245.09	6.19×10^{-3}	**2.36**	252.70	6.20×10^{-5}
	I-B	12	22.15	179.34	4.83	219.27	5.21×10^{-7}	5.22	218.36	1.65×10^{-2}	**3.00**	223.47	1.61×10^{-5}
0.6	I-C	30	29.54	230.25	5.89	307.54	1.41×10^{-9}	5.19	309.85	1.43×10^{-3}	**2.63**	318.20	3.56×10^{-7}
	I-D	12	20.18	232.11	4.28	278.34	5.40×10^{-8}	4.73	277.03	1.78×10^{-2}	**2.71**	282.91	8.94×10^{-6}
	I-E	6	11.82	287.65	3.12	316.02	8.78×10^{-8}	3.48	314.85	2.63×10^{-2}	**2.25**	318.84	7.78×10^{-5}
	I-F	3	4.29	296.71	1.43	305.56	5.35×10^{-7}	1.91	304.09	1.05×10^{-3}	**1.13**	306.50	1.72×10^{-4}
	I-G	30	24.17	531.23	5.18	664.26	2.06×10^{-9}	5.00	665.56	1.31×10^{-1}	**2.42**	683.65	2.75×10^{-5}
	I-H	12	15.06	535.09	2.95	611.43	2.77×10^{-8}	3.34	608.92	9.63×10^{-3}	**1.80**	618.66	2.49×10^{-4}
	I-4	20	32.30	95.98	5.56	133.91	1.71×10^{-14}	4.54	135.36	2.39×10^{-7}	**2.40**	138.39	2.27×10^{-9}
	I-5	20	34.33	96.27	6.20	137.51	5.07×10^{-16}	6.12	137.62	2.93×10^{-1}	**2.76**	142.55	8.79×10^{-10}
	I-6	8	19.63	98.21	3.14	118.37	3.63×10^{-6}	3.38	118.08	1.62×10^{-1}	**2.08**	119.66	6.08×10^{-4}
	I-A	20	32.95	140.65	5.88	197.46	1.94×10^{-7}	6.12	196.95	1.14×10^{-1}	**2.92**	203.66	1.14×10^{-5}
	I-B	8	21.41	141.44	4.02	172.76	5.61×10^{-7}	4.84	171.27	3.75×10^{-3}	**2.48**	175.53	1.48×10^{-4}
0.4	I-C	20	31.50	179.73	5.50	247.96	2.99×10^{-8}	5.77	247.24	1.12×10^{-1}	**2.66**	255.41	5.49×10^{-7}
	I-D	8	19.61	180.87	3.74	216.59	1.92×10^{-6}	4.48	214.91	2.07×10^{-2}	**2.56**	219.23	2.34×10^{-6}
	I-E	4	9.00	222.76	2.14	239.55	1.89×10^{-8}	2.64	238.34	1.59×10^{-2}	**1.64**	240.78	1.21×10^{-6}
	I-F	2	1.87	229.63	1.72	229.97	1.22×10^{-1}	1.97	229.39	2.48×10^{-2}	**1.55**	230.37	8.79×10^{-3}
	I-G	20	25.35	405.93	4.45	519.61	5.35×10^{-9}	4.65	518.51	2.74×10^{-2}	**2.32**	531.16	2.91×10^{-5}
	I-H	8	14.64	408.88	3.15	463.92	2.75×10^{-9}	3.52	462.16	2.56×10^{-2}	**1.87**	470.06	4.82×10^{-5}

Averages from 30 runs. Multi-Start generates random solutions and chooses the best. RR-GA is a genetic algorithm with random repair. FP-GA is a genetic algorithm with FP normalization. OPT-GA is a genetic algorithm with OPT normalization. [1] The one-tailed t-test with the null hypothesis of RR-GA = Multi-Start. [2] The one-tailed t-test with the null hypothesis of FP-GA = RR-GA. [3] The one-tailed t-test with the null hypothesis of OPT-GA = FP-GA.

5. Conclusions

We present the maximum k-coverage problem (MKCP) and analyze its representation and solution space. If we apply a GA to the MKCP, then we immediately encounter the issue of redundancy in the genotype space, which is larger than the phenotype space that we characterize as a quotient space. We introduce a method of normalizing chromosomes that ensures a crossover produces feasible offspring with genes that are column vectors of the MKCP matrix and are as dissimilar as possible. This normalization was implemented using the Hungarian method [40]. We performed experiments which showed the effectiveness of this approach.

In this study, we adopted two locus-based metrics of the discrete metric and its extended version derived from Hamming distance between genes (column vectors). However, other metrics such as some variants of Cayley metric on permutations [51,52] may also be applied to the proposed theoretical framework. In the case of such non-locus-based metric, we should design a new crossover tailored to the metric. This investigation will be a promising work, which we leave for future study.

As mentioned in the Introduction, the proposed theoretical framework can be applied to real-world applications such as the cyber-physical social systems and public safety networks. We leave this applied work for future study. We also expect that this approach can be applied to other problems which have the same representation for their solutions. By expanding this technique to solution representations of variable length as in [53,54], we believe it could also be applied to the set cover problem.

Author Contributions: Conceptualization, Y.Y. and Y.-H.K.; methodology, Y.-H.K.; software, Y.Y.; validation, Y.Y.; formal analysis, Y.Y.; investigation, Y.Y.; resources, Y.Y.; data curation, Y.Y.; writing—original draft preparation, Y.Y.; writing—review and editing, Y.-H.K.; visualization, Y.Y.; supervision, Y.-H.K.; project administration, Y.-H.K.; and funding acquisition, Y.-H.K. All authors have read and agreed to the published version of the manuscript.

Funding: The present research was conducted by the Research Grant of Kwangwoon University in 2020. This research was a part of the project titled 'Marine Oil Spill Risk Assessment and Development of Response Support System through Big Data Analysis' funded by the Korea Coast Guard. This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science, ICT & Future Planning) (No. 2017R1C1B1010768).

Acknowledgments: The authors thank Byung-Ro Moon for his valuable suggestions, which improved this paper.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Disclosure: A preliminary version of this paper appeared in the Proceedings of the Genetic and Evolutionary Computation Conference, pp. 593–598, 2008. In comparison with the conference paper, this paper was newly rewritten with the following new materials: (i) new work: complete literature survey (Section 1) and complete theoretical work of the proposed normalization (Sections 2 and 3.2); and (ii) improved work: an improved genetic algorithm (through the improvement of normalization technique), and its largely-extended experiments together with their statistical verification (Section 4).

References

1. Hochbaum, D.S.; Pathria, A. Analysis of the greedy approach in problems of maximum k-coverage. *Nav. Res. Logist.* **1998**, *45*, 615–627. [CrossRef]
2. Caprara, A.; Fischetti, M.; Toth, P.; Vigo, D.; Guida, P.L. Algorithms for railway crew management. *Math. Program.* **1997**, *79*, 125–141. [CrossRef]
3. Indyk, P.; Mahabadi, S.; Mahdian, M.; Mirrokni, V.S. Composable Core-sets for Diversity and Coverage Maximization. In Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Snowbird, UT, USA, 22–27 June 2014; pp. 100–108.
4. Indyk, P.; Vakilian, A. Tight Trade-offs for the Maximum k-Coverage Problem in the General Streaming Model. In Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 200–217.
5. Saha, B.; Getoor, L. On Maximum Coverage in the Streaming Model & Application to Multi-topic Blog-Watch. In Proceedings of the the SIAM International Conference on Data Mining, Sparks, NV, USA, 30 April–2 May 2009; pp. 697–708.

6. Zheng, S.; Dmitriev, P.; Giles, C.L. Graph Based Crawler Seed Selection. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, Madrid, Spain, 20–24 April 2009; pp. 1089–1090.
7. Chierichetti, F.; Kumar, R.; Tomkins, A. Max-cover in Map-reduce. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 231–240.
8. Li, F.H.; Li, C.T.; Shan, M.K. Labeled Influence Maximization in Social Networks for Target Marketing. In Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing, Boston, MA, USA, 9–11 October 2011; pp. 560–563.
9. Hammar, M.; Karlsson, R.; Nilsson, B.J. Using Maximum Coverage to Optimize Recommendation Systems in e-Commerce. In Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; pp. 265–272.
10. Yoon, Y.; Kim, Y.H. An Efficient Genetic Algorithm for Maximum Coverage Deployment in Wireless Sensor Networks. *IEEE Trans. Cybern.* **2013**, *43*, 1473–1483. [CrossRef] [PubMed]
11. Yaghini, M.; Karimi, M.; Rahbar, M. A set covering approach for multi-depot train driver scheduling. *J. Comb. Optim.* **2015**, *29*, 636–654. [CrossRef]
12. Liu, Q.; Cai, W.; Shen, J.; Fu, Z.; Liu, X.; Linge, N. A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. *Secur. Commun. Netw.* **2016**, *9*, 4002–4012. [CrossRef]
13. Máximo, V.R.; Nascimento, M.C.V.; Carvalho, A.C.P.L.F. Intelligent-guided adaptive search for the maximum covering location problem. *Comput. Oper. Res.* **2017**, *78*, 129–137. [CrossRef]
14. Tsiropoulou, E.E.; Thanou, A.; Papavassiliou, S. Quality of Experience-based museum touring: A human in the loop approach. *Soc. Netw. Anal. Min.* **2017**, *7*, 33. [CrossRef]
15. Sikeridis, D.; Tsiropoulou, E.E.; Devetsikiotis, M.; Papavassiliou, S. Socio-spatial resource management in wireless powered public safety networks. In Proceedings of the IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018, pp. 810–815.
16. Fragkos, G.; Tsiropoulou, E.E.; Papavassiliou, S. Disaster Management and Information Transmission Decision-Making in Public Safety Systems. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019, pp. 1–6.
17. Garey, M.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: San Francisco, CA, USA, 1979.
18. Caserta, M.; Doerner, K.F. Tabu search-based metaheuristic algorithm for the large-scale set covering problem. In *Metaheuristics: Progress in Complex Systems Optimization*; Springer: New York, NY, USA, 2007; pp. 43–63.
19. Aickelin, U. An indirect genetic algorithm for set covering problems. *J. Oper. Res. Soc.* **2002**, *53*, 1118–1126. [CrossRef]
20. Beasley, J.E.; Chu, P.C. A Genetic Algorithm for the Set Covering Problem. *Eur. J. Oper. Res.* **1996**, *94*, 392–404. [CrossRef]
21. Balaji, S.; Revathi, N. A new approach for solving set covering problem using jumping particle swarm optimization method. *Nat. Comput.* **2016**, *15*, 503–517. [CrossRef]
22. Al-Shihabi, S.; Arafeh, M.; Barghash, M. An improved hybrid algorithm for the set covering problem. *Comput. Ind. Eng.* **2015**, *85*, 328–334. [CrossRef]
23. Ausiello, G.; Boria, N.; Giannakos, A.; Lucarelli, G.; Paschos, V. Online maximum *k*-coverage. *Discret. Appl. Math.* **2012**, *160*, 1901–1913. [CrossRef]
24. Yu, H.; Yuan, D. Set coverage problems in a one-pass data stream. In Proceedings of the the SIAM International Conference on Data Mining, Austin, TX, USA, 2–4 May 2013, pp. 758–766.
25. Chandu, D.P. Big Step Greedy Heuristic for Maximum Coverage Problem. *Int. J. Comput. Appl.* **2015**, *125*, 19–24.
26. Wang, Y.; Ouyang, D.; Yin, M.; Zhang, L.; Zhang, Y. A restart local search algorithm for solving maximum set *k*-covering problem. *Neural Comput. Appl.* **2018**, *29*, 755–765. [CrossRef]
27. Lin, G.; Guan, J. Solving maximum set *k*-covering problem by an adaptive binary particle swarm optimization method. *Knowl.-Based Syst.* **2018**, *142*, 95–107. [CrossRef]
28. Yoon, Y.; Kim, Y.H.; Moon, B.R. Feasibility-Preserving Crossover for Maximum *k*-Coverage Problem. In Proceedings of the Genetic and Evolutionary Computation Conference, Atlanta, GA, USA, 12–16 July 2008; pp. 593–598.

29. Fraleigh, J.B. *A First Course in Abstract Algebra*, 7th ed.; Addison Wesley: Boston, MA, USA, 2002.
30. Burago, D.; Burago, Y.; Ivanov, S.; Burago, I.D. *A Course in Metric Geometry*; American Mathematical Society: Providence, RI, USA, 2001.
31. Yoon, Y.; Kim, Y.H.; Moraglio, A.; Moon, B.R. Quotient geometric crossovers and redundant encodings. *Theor. Comput. Sci.* **2012**, *425*, 4–16. [CrossRef]
32. Choi, S.S.; Moon, B.R. Normalization in Genetic Algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, IL, USA, 12–16 July 2003; pp. 862–873.
33. Dorne, R.; Hao, J.K. A New Genetic Local Search Algorithm for Graph Coloring. In Proceedings of the Fifth Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands, 27–30 September 1998; pp. 745–754.
34. Laszewski, G. Intelligent Structural Operators for the *k*-way Graph Partitioning Problem. In Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, CA, USA, 13–16 July 1991; pp. 45–52.
35. Mühlenbein, H. Parallel genetic algorithms in combinatorial optimization. In *Computer Science and Operations Research: New Developments in Their Interfaces*; Pergamon: Oxford, MS, USA, 1992; pp. 441–456.
36. Van Hoyweghen, C.; Naudts, B.; Goldberg, D.E. Spin-flip symmetry and synchronization. *Evol. Comput.* **2002**, *10*, 317–344. [CrossRef]
37. Kang, S.J.; Moon, B.R. A Hybrid Genetic Algorithm for Multiway Graph Partitioning. In Proceedings of the Genetic and Evolutionary Computation Conference, Las Vegas, NV, USA, 8–12 July 2000, pp. 159–166.
38. Moraglio, A.; Kim, Y.H.; Yoon, Y.; Moon, B.R. Geometric Crossovers for Multiway Graph Partitioning. *Evol. Comput.* **2007**, *15*, 445–474. [CrossRef]
39. Choi, S.S.; Moon, B.R. Normalization for Genetic Algorithms with Nonsynonymously Redundant Encodings. *IEEE Trans. Evol. Comput.* **2008**, *12*, 604–616. [CrossRef]
40. Kuhn, H.W. The Hungarian Method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
41. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1955.
42. Avis, D. A survey of heuristics for the weighted matching problem. *Networks* **1983**, *13*, 475–493. [CrossRef]
43. Beasley, J.E. OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **1990**, *41*, 1069–1072. [CrossRef]
44. Eshelman, L.J. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*; Morgan Kaufmann: Burlington, MA, USA, 1991; pp. 265–283.
45. Alba, E.; Luque, G.; Araujo, L. Natural language tagging with genetic algorithms. *Inf. Process. Lett.* **2006**, *100*, 173–182. [CrossRef]
46. Cordón, O.; Damasb, S.; Santamaría, J. Feature-based image registration by means of the CHC evolutionary algorithm. *Image Vis. Comput.* **2006**, *24*, 525–533. [CrossRef]
47. Guerra-Salcedo, C.; Whitley, D. Genetic Search for Feature Subset Selection: A Comparison Between CHC and GENESIS. In Proceedings of the Third Annual Conference on Genetic Programming; Morgan Kaufmann: Burlington, MA, USA, 1998; pp. 504–509.
48. Nebro, A.J.; Alba, E.; Molina, G.; Chicano, F.; Luna, F.; Durillo, J.J. Optimal antenna placement using a new multi-objective CHC algorithm. In Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, UK, 7–11 July 2007; pp. 876–883.
49. Tsutsui, S.; Goldberg, D.E. Search space boundary extension method in real-coded genetic algorithms. *Inf. Sci.* **2001**, *133*, 229–247. [CrossRef]
50. Yoon, Y.; Kim, Y.H.; Moraglio, A.; Moon, B.R. A Theoretical and Empirical Study on Unbiased Boundary-extended Crossover for Real-valued Representation. *Inf. Sci.* **2012**, *183*, 48–65. [CrossRef]
51. Pinch, R.G.E. The distance of a permutation from a subgroup of S_n. *arXiv* **2005**, arXiv:math/0511501.
52. Kim, Y.H.; Moraglio, A.; Kattan, A.; Yoon, Y. Geometric generalisation of surrogate model-based optimisation to combinatorial and program spaces. *Math. Probl. Eng.* **2014**, *2014*, 184540. [CrossRef]

53. Nam, Y.W.; Kim, Y.H. Automatic jazz melody composition through a learning-based genetic algorithm. In Proceedings of the 8th International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART), Leipzig, Germany, 24–26 April 2019; Lecture Notes in Computer Science. Springer: Berlin, Germany; Volume 11453, pp. 217–233.
54. Lee, J.; Kim, Y.H. Epistasis-based basis estimation method for simplifying the problem space of an evolutionary search in binary representation. *Complexity* **2019**, *2019*, 2095167. [CrossRef]