Designing a metaheuristic mining algorithm to separate two-color points in a two-dimensional environment

Diseño de un algoritmo de minería metaheurístico para separar puntos de dos colores en un entorno bidimensional

Author:

Parisa Aghazade[1,*]
Alireza Bagheri[2]
Mohamadmansoor Riahi Kashani[3]

SCIENTIFIC RESEARCH

How to cite this paper:
Aghazade. P., Bagheri. A., Kashani. M. R. Designing a metaheuristic mining algorithm to separate two-color points in a two-dimensional environment. Tehran, Iran. Innovaciencia. 2019; 7 (2): 1-14. DOI: http://dx.doi.org/10.15649/2346075X.773

Reception date:
Received: 13 February 2019
Accepted: 28 April 2019
Published: 25 October 2019

Keywords:
Data-mining, Separation of points, Computational geometry, metaheuristic Algorithm, Simulated Annealing (SA).

ABSTRACT

The separation of color points is one of the important issues in computational geometry, which is used in various parts of science; it can be used in facility locating, image processing and clustering. Among these, one of the most widely used computational geometry in the real-world is the problem of covering and separating points with rectangles. In this paper, we intend to consider the problem of separating the two-color points sets, using three rectangles. In fact, our goal is to separate desired blue points from undesired red points by three rectangles, in such a way that these three rectangles contain the most desire points. For this purpose, we provide a metaheuristic algorithm based on the simulated annealing method that could separates blue points from input points, , in time order O (n) with the help of three rectangles. The algorithm is executed with C# and also it has been compared and evaluated with the optimum algorithm results. The results show that our recommended algorithm responses is so close to optimal responses, and also in some cases we obtains the exact optimal response.

* Corresponding author:
1 MSC student of Islamic Azad University, North Tehran Branch, Tehran, Iran. e-mail: Parisa.Aghazade472@gmail.com
2 Assistant professor and faculty member of Amirkabir University
3 Assistant professor and faculty member, Islamic Azad University, North Tehran University, Tehran, Iran

1. INTRODUCTION

In computational geometry, a geometric model is presented as a branch of design and analysis. This branch of science is actually a combination of geometry and computer that has many uses in various majors such as computer graphics, geographic information systems, integrated circuit design, statistics, machine vision and robotics [1].

In this major, the separation of color points is considered as one of the most important and widely used computational geometry.

So far, numerous studies have been carried out with different shapes such as circle, rectangle and square [2]. In this study, we focused on the separation of color points with three rectangles. In order to solve this problem, we intend to introduce a metaheuristic algorithm based on Simulated Annealing (SA) algorithm, evaluating its results and comparing it with the optimal algorithm.

2. STATEMENT OF THE RESEARCH PROBLEM

In the covering problems, a set of geometric objects such as a point, a line segment, and a circle are covered by one or more geometric objects such as a circle, rectangle, square, and a triangle. As shown, in Figure 1, covering the points of a page is represented by a rectangle.

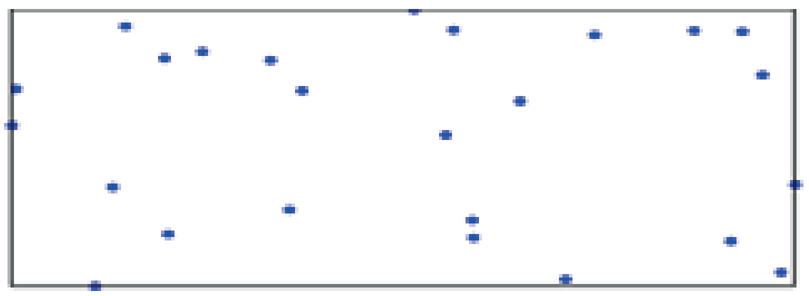


Figure 1: covering the page points by a rectangle

From a basic perspective, it is assumed that all given points are indistinguishable from each other. Consequently, the purpose of this view is to cover the geometric shape in all directions so that, the criterion such as the area or perimeter of the shape become minimum. This issue is known as the full coverage issue, but due to limitation of resources in the real-world, full coverage may not always be possible. According to these cases, the problem of full coverage and optimal coverage by different geometric shapes has been widely studied [3].

In another perspective on the coverage problems, it is assumed that the points are not of the same quality. More precisely, in this view it is assumed that some of the given points are desirable and the rest of the points are undesirable. Therefore, in this case, the goal is to separate the desired and undesired points from each other. This approach is known as the separation problem, and the geometric shape in this matter is called the separator.

It is assumed that there are a set of points with two different colors on the page, so that the points with a certain color, representing the desired points and points with other colors, represent undesired points [4]. In Figure 2, the separation of blue and red points is represented.

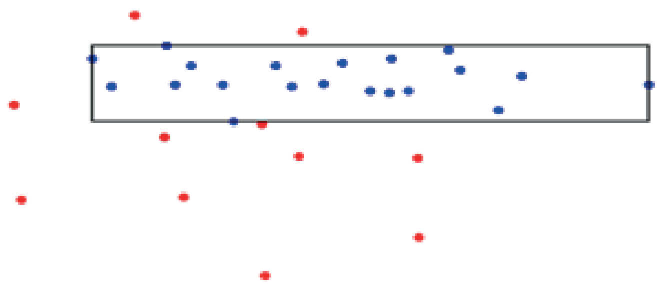


Figure 2: separating two-color points by a rectangle

A variety of research has been done to separate two-color points using a rectangle. The blue and red points may be located on the page, which their complete separation is not possible by a rectangle or two rectangles and therefore it is necessary to use three rectangles.

The issues related to the separation of points with geometric shapes, have many diversity and they were shown in a completely different issue with just a slightest change. Examining the largest or smallest rectangle separator, finding the largest empty rectangle of points, separating points with more than two colors, and separating points with more than one rectangle, are some of these issues. In this matter, it is assumed that there are a number of points with two colors of blue and red, in the 2d space of R2, and the main goal is to find the smallest three rectangles that can cover the most blue points in this space. In other words, all the blue points may either be located inside the rectangle or at the boundaries of these three rectangles, while all the red points are outside of the three rectangles or located at the boundaries. Figure 3 shows three rectangles covering the blue points.

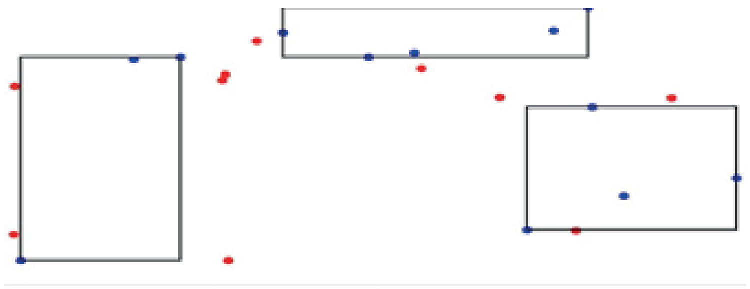


Figure 3: the three rectangles contains blue points as much as possible, and red points are outside them.

3. DESCRIPTION OF THE SIMULATED ANNEALING (SA)

Simulated Annealing Algorithm is a simple and effective metaheuristic optimization algorithm for solving optimization problems in large search domains. In fact, this Algorithm applies to issues that finding an approximate response for general optimization, is more important than finding a precise response to local optimum at a limited and specific time [5]. The gradual Annealing technique is used by metallurgists to achieve a state in which the solid is well-developed and its energy is minimized. The goal is to maximize crystalline size in the solid state of the Annealing substance. This technique involves placing the material at high temperature and then gradually reducing the aforementioned temperature. In fact, gradual and slow Annealing in this algorithm can be considered as a gradual reduction in the possibility of choosing worse responses when searching in the response space. In Figure 4, the general schema has been shown.

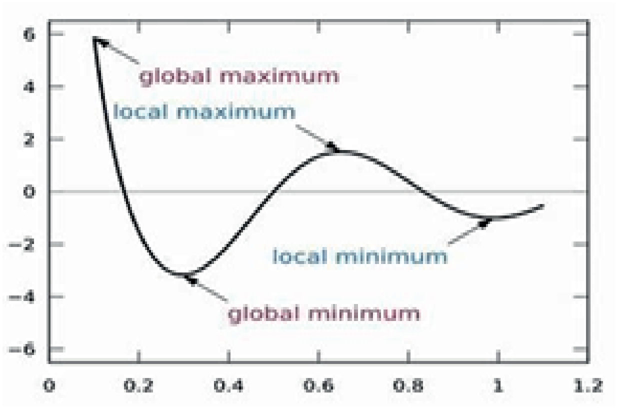


Figure 4: The schema of a region includes peaks and valleys where the ball can leap into different parts.

In this case, the goal is to place the ball in the deepest valley (highest level of energy) of the area shown in Figure 4, but the location is known. To do this, first we allow the ball to be thrown into different areas with a high radius. This is done in SA by choosing large values of the neighborhood radius. The ball after throwing, naturally will go down to the areas containing the valley; But since this may not be the deepest valley, then the ball must be allowed to be mutated from this state and be thrown again.

This will continue to make sure that the ball is finally at the deepest part of the valley for once. To keep the ball in the desired area, we must decrease the ball's permission to throw over time. Also, the probability of a ball falling to areas with higher heights (out of the valley) must reduce.

Similarly, in the SA algorithm, the radius of the neighborhood is initially large and, as a result, different values of the solution space were chosen. Also, by choosing a high temperature parameter, it is also possible to go to high energy states, but over time, the neighboring radius is reduced, and as a result, the selected values of the response space are slightly different than each other, so by decreasing the temperature, the probability of going to higher energy states is reduced. Consequently, in the initial steps, by choosing a large neighborhood radius and the probability of going up to the worse moves, the probability of getting caught in the maximum or minimum points of the locality decreases. Also, in the lower steps, with decrease in the neighboring radius, and considering that all the points were randomly searched (a number of points are in their optimal points, or close to it) they reach a general optimal point and stay in it. In general, metaheuristic SA algorithm uses neighbors with same answers, to search for the space of the answers. This algorithm can achieve the best possible answer by spending a sufficient time.

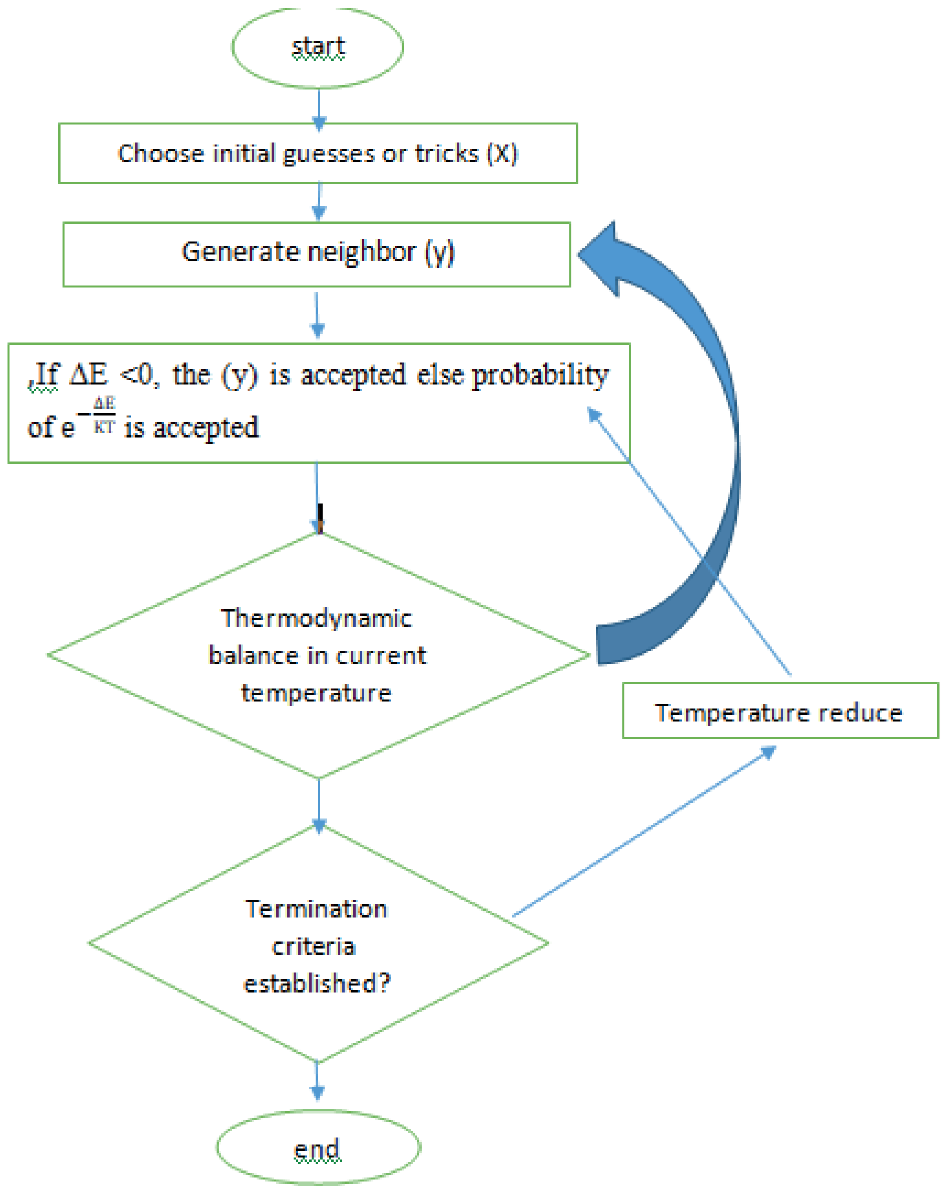
The probability of a transition from the current state such as **s,** to a new candidate state, such as as $P(e.e'.T)$, is also determined by a probability function as in which e = E (s) and e' = E (S′) (The E function is the energy state of the space and T represents the variable temperature with the system time).

Lower energies are better than those with higher energies. Also the probability function of P should be positive, even when e is smaller than e'. This feature ensures that the algorithm does not get caught in a local response.

When T asks for zero, the P probability must either be zero (e smaller than e') or be a positive number (e' smaller than e). For values that are enough smaller than T, the system moves to the point of minimum energy. It should be noted that by placing T = 0, the issue is trimmed into a greedy algorithm, and its moves will only be towards the points with less energy.

In the initial case of Simulated Annealing (SA), when e' is smaller than e, the probability $P(e.e^{'}.T)$ is equal to 1, which means that the procedure always moves to lower points, independently from temperature. Although this condition is not required for the operation of the method, many of its simulation and implementations are considered as part of the definition of the method. The function P is always chosen so that the probability of accepting a motion (when the difference between two modes is low) is reduced. For example, small upward moves are more likely than large moves. With these status, it's clear that temperature plays a crucial role in controlling changes in the system (due to its sensitivity to energy changes). More precisely, in large T values, changes in system state are more sensitive to larger changes in the system rather than when the temperature is smaller.

The overall structure of the Simulated Annealing algorithm and its stages are briefly shown in Figure 5. As shown, the SA algorithm has two repetition chains, the first chain responsible for finding local responses at any temperature and the second chain is responsible for the reduction of temperature.



start
Choose initial guesses or tricks (X)
Generate neighbor (y)
,If ΔE <0, the (y) is accepted else probability of $e^{-\frac{\Delta E}{KT}}$ is accepted
Thermodynamic balance in current temperature
Temperature reduce
Termination criteria established?
end

Figure 5: Flowchart Simulated Annealing algorithm.

The flowchart shows the simplest form of the SA algorithm and its stages are as follows:

- Choose initial guesses or tricks (X) randomly.
- Generate random new answer or neighbor's answer (Y) from the initial answer.
- Energy comparison of initial response (X) with neighbor's response (Y).

If the energy difference is negative ($\Delta E < 0$), or in other word the energy of the new answer (Y) is less and better than the initial answer (X), then the answer (Y) is accepted as the current answer; otherwise, if the answer (Y) is More and worse than the initial answer (X), then the answer (Y) is accepted as probable. It means that, first, a random number is generated between zero and one; if this number is smaller and better than $e^{-\frac{\Delta E}{KT}}$, the new answer (Y) is selected; Otherwise it is not accepted and the answer of a new neighbor is selected. If the algorithm has reached the final temperature, it goes to the next stage, otherwise it remains at the same temperature and tries to find the next answer by re-processing from the current answer. It ends, when the conditions and criteria for termination of the algorithm are established; otherwise, the temperature is reduced and a new answer is produced again.

4. RESEARCH BACKGROUND

The issue of finding the minimum circle, was raised by Sylvester in 1857 [6]. The most basic method solve the problem by examining all possible states at time O (n^4). Over time, algorithms were proposed to resolve this problem, all of which tried to reduce the execution time of this algorithm. In 1975, an algorithm with time order O (nlogn) was proposed to solve this problem and then proceeded to optimum time O (n), with the help of a linear programming method [7]. The problem is to find the smallest rectangle parallel to the x-axis, which covers exactly **n** points of **P**, and solved by the agraval [8] at time $O(k^2 \text{nlogn})$, Epstein [9] at time $O(n^2)$ and Segal M. and Kedem K. [7] At time $O(n+k\ (n\text{-}k)^2)$.

In 1994 optimal cover was proposed by Epstein and Arisan, using a fixed-size rectangular and coordinate axes [10] and presented algorithmic with time order $O(k^2 \text{nlogn})$ to solve this problem.

In another version of the cover problem, which is called optimum coverage or more precisely called maximal coverage, it is assumed that a set of **n** points in a page and a fixed-dimensional geometry is given and the goal is to put this shape on the page in a way that covers maximum number of points. This problem has been proposed using different geometric shapes; Problems such as minimum area enclosing using a circle [11], minimum area enclosing using a fixed-dimensional rectangle and coordinate axes [12] and optimal coverage using a convex polygon [13].

In 2009, the problem of covering the points with two Isothetic rectangles was defined by Saha and Dos [5]. A set of **n** points in the page is covered by two Isothetic rectangles at any arbitrarily orientation. This algorithm, at the time $O(n^3)$ can be used to solve another well-known optimization problem. This issue is to cover points with two Isothetic rectangles, which gain the minimum area of the two rectangles at any arbitrarily orientation.

Eckstein et al. [12] examined the issue of separation in a set of points with a rectangle, so that none of the red points located inside the separators rectangle. In addition, the number of blue points inside the rectangular cover is maximized. They showed that this is an **NP-hard** problem.

In 2013, Afsane Halatabadi Farahani [1] presented a heuristic algorithm for separation of color points with two circles, which answers at the time of O (n). In same year, Sarah Khalafi et al [13] proposed a solution to separate points with the largest single-color polygon were at time O (nlogn) the separation of points will be done.

In the year 2012, Zahra Moslehi et al. [14] proposed an algorithm for separating the set of two-color points with two indistinguish and isothetic rectangles. They took two sets of blue and red points with a total size of **n** in order to report all orientation of the two isothetic separator with empty and non-empty intersections, so that all the blue points were located inside the rectangles or all the red points were located outside them. These two states are solved with time order O ($n^2(n)$) and O (n^2 *logn*) respectively.

5. THE PROPOSED METHOD

In this section the proposed algorithm were outlined in order to separate the two blue and red points with three rectangles. This algorithm is not limited to the number of rectangles and can be used to solve various problems, arbitrarily. the assume is that the blue and red points are located in a given two-dimensional (2d) space, and dispersed in this space without any restrictions on the location. Figure 6 shows an example of dispersion of two-color points in 2d space.

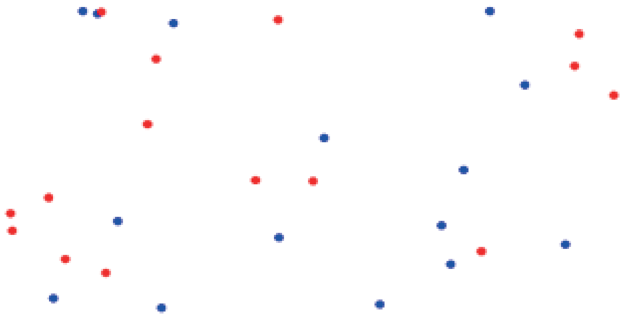
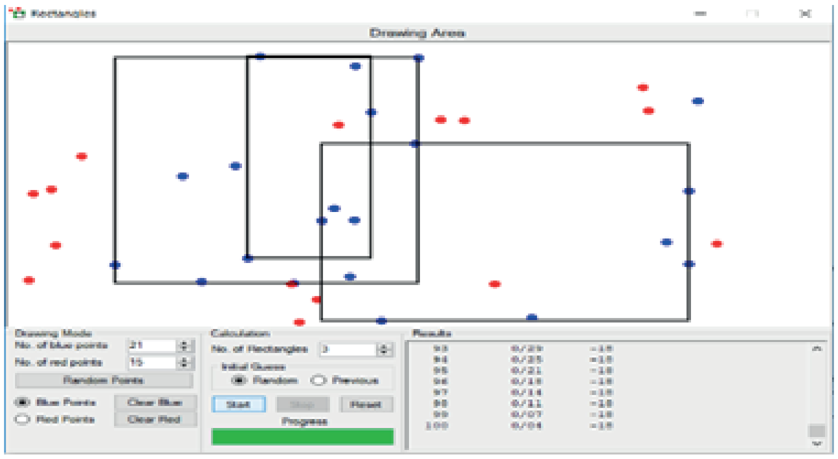


Figure 6: Probable mode of dispersion of two-color points in 2d space.

The points that are located inside the area are considered as internal points, and points that are not inside the area of any rectangle, are considered as external points. Rectangles are not necessarily separate and may overlap (Fig 7).



No. of blue points 21
No. of red points 15
No. of Rectangles 3

```
93      0/29     -18
94      0/25     -18
95      0/21     -18
96      0/18     -18
97      0/14     -18
98      0/11     -18
99      0/07     -18
100     0/04     -18
```

Figure 7: Deploying of three rectangles with overlapping regions in two-dimensional space

The Desired Configuration of algorithm is placement of the maximum number of blue points and the minimum number of red points in the rectangle, so we need to define an Objective function, which, our desirable mode is the minimum of that (Formula 1) and use a minimizing algorithm of Objective function. In this research we have used the Simulated Annealing algorithm (SA). The system energy (E) is calculated in a given state (s) as follows:

$$E(s) = n_{Red} - n_{Blue} \tag{1}$$

Where n_{Red} and n_{Blue} are the number of red points and the number of points in the rectangles, respectively. Therefore, the optimal mode is **s*** that the rectangles contain most of the blue points and the minimum points of the red, which corresponds to the minimum energy of the system. If the blue points are equal to **n** and the number of rectangles is **m=3**, and each of the blue points can be located in one of the following 4 states (**m + 1 = 4**):

- state 0: not in any rectangle
- state 1: Inside rectangular No. 1
- state 2: Inside the rectangle No. 2
- state 3: Inside the rectangle No.3

Each blue point is randomly positioned in one of the states and by storing the position of each point in the rectangular data structure, we consider the maximum and minimum values of **x** and **y** of the blue points and draw the sides of the rectangles. In fact, rectangles are drawn by blue points. Therefore, the system state (**s**) can be shown by an array with length of **n** representing numbers 0 to **m**, that **m** indicating the number of rectangles. For example, if there are 10 blue points (**n = 10**), the state of the system can be considered as arrays in table 1. This array from left to right shows that point number 1 is not inside any rectangles. Points 2, 4, and 10 are in rectangle 1; points 5, 6 and 9 in rectangle 2 and points 3 and 7 are also in rectangle 3. As a result, this array can be used to generate different states of the system.

Table 1: a possible state (s) for 10 blue points and 3 rectangles

0	**1**	**3**	**1**	**2**	**2**	**3**	**0**	**2**	**1**

In the next step, the red point's status is checked with rectangles, and by comparing the coordinates of the red points with the coordinates of the rectangle, it is determined that which points are inside the rectangles and which ones are outside. Finally, the system's energy is obtained by calculating the difference of red points inside the rectangles with blue points inside the rectangles (Formula 1). If there are fewer red points inside the rectangle, the value of the Objective function becomes lower and closer to the optimal value (in some cases the optimal response is achieved).

To move from the current state of system to a neighboring state, in accordance with simulated annealing theory, the changes should be gradual and irreversible. To create a new neighbor from the current state, we first select a point randomly, then change the value of this point randomly; and by changing the position of the points on the page, the new rectangle will be traced by the new coordinates of the blue points. Table 2 shows a random and reversible change to generate a new neighbor. In this change, the fourth point is assumed to be outside of all rectangles.

Table 2: Random and reversible Change to generate a new neighbor.

0	**1**	**3**	**1**	**2**	**2**	**3**	**0**	**2**	**1**

$$\downarrow$$

0	**1**	**3**	**0**	**2**	**2**	**3**	**0**	**2**	**1**

After the states of all points are identified, the points in a particular rectangle define the rectangle's boundaries. That is, the leftmost, rightmost, highest and lowest points in each rectangle, indicate the boundaries of the left, right, up and down, respectively. The coordinates of the red points are also examined with new rectangles, and it is determined that each one is located inside or outside of the rectangle. Finally, the system's Objective function is recalculated and the differences in red and blue points are obtained. If the energy of neighboring state, is less than energy of the initial system state, the new neighboring state is considered as the current mode; otherwise, the algorithm accepts the answer with the probability of $(-\frac{\Delta E}{T})$, as the current answer. In this regard, the $\boldsymbol{\Delta E}$ is the difference between the Objective function of the current answer and the answer of the neighbor; also the T represents the temperature. At each temperature, several repetitions are performed, and then the temperature is gradually reduced. In the initial steps, the temperature is very high, so it is more likely to accept worse answers. With the gradual reduction of temperature in the final steps, there is less chance to accept worse answers, and so the algorithm converges to an optimal response. The P seudo-code for our proposed algorithm, which is executed based on the simulated annealing algorithm, is shown in Table 3.

Table 3: Steps of the proposed algorithm

begin	1
Coordinates of the blue and red points are stored	2
Selecting Primary Guess: The initial guess in array with length of **n** from the set of numbers S_current = {0.1.2.···.m} where **m** is the number of rectangles and **n** is the number of blue points. The zero number in the **i-th** array indicates that the **i-th** point is not in any rectangle, and the number **j** indicates that it is in the rectangle **j**. The initial temperature of the system is determined as $T_0=0.1*n$, where **n** is the total number of points on the page.	3
The Objective function (System energy) is calculated: **E_current** To calculate the Objective function, the blue and red points condition is checked and the number of red and blue points inside the rectangles, is counted. The target function is calculated using Formula 1. The best Objective function is obtained (**E* = E_current**) and the corresponding state is stored: **S* = S_current**	4
To generate the new neighbor state, (**S_new**), one of the array indexes from (1.2.···.**n**) is randomly selected and changed in random form. Of course, to generate a neighboring state, the blue point does not change, and only its status changes and is randomly determined to be outside of the rectangles or inside other rectangles. In the neighboring state, one of the rectangles is changed, so the neighbor state is the same as the previous one.	5
The Objective function (System energy), **E_new** is calculated for the new random state. To calculate the Objective function of neighboring mode, the status of blue and red points is checked and the number of red and blue points is counted inside the rectangles.	6
Accepting new state will be checked as follows: The energy difference $\Delta E = E_{new}-E_{current}$ is calculated and accepted by the following conditions: If $\Delta E < 0$, the new state is accepted. If $\Delta E > 0$, the new state is accepted with the probability of $\exp(-\frac{\Delta E}{T})$. For this purpose, a random number is generated with a uniform distribution in the interval (0.1); and if it was less than the $\exp(-\frac{\Delta E}{T})$, the new state will accepts. The higher the value of ΔE, the less likely to accept the state. Also, by reducing the temperature, the chance of accepting the state becomes less.	7
If the new state is adopted, the energy and current state will be updated: $E_{current}=E_{new}$ ؛ $S_{current}=S_{new}$ To find the best answer, the condition $\mathbf{E_{current}<E^*}$ is checked and if it is established $\mathbf{E^*=E_{current}}$ و $\mathbf{S^*=S_{current}}$ are considered.	8
Temperature reduction: The SA algorithm is simulated by reducing the temperature and gradually annealing the system. A gradual annealing is considered as linear reduction with linear rate: **T=T-δT**	9
Convergence condition test: If the Objective function reaches the minimum possible value, the algorithm has reached the final answer and it goes to the next stage. Stop condition check: The algorithm goes to the next step if it reaches the maximum number of iterations. The maximum iterations in the SA algorithm are also equivalent to reaching the minimum temperature. If the $T < T_{min}$ it shows that the system has cooled over that it can change the situation; and the condition is set to stop. In case of not reaching one of the above conditions, the algorithm will go to the next stage, otherwise it will also go to step 5.	10
The best answer found means that **S*** (corresponding to the lowest value of the objective function, **E***) is reported.	11
the end	12

In order to execute and test the proposed algorithm, we developed a software under Windows operating system and C #. The evaluation results is described in the next section.

6. EVALUATION AND ANALYSIS OF TESTS

In this section, experiments with different input values are executed on the algorithm and the results are examined and compared with the optimal answer. In order to calculate the recommended algorithm score and the optimal algorithm score, 1000 runs of the program with different input values and various number of points, had been done and results were recorded. Since our recommended algorithm has a random nature and receives a different response with each run, so in order to gain a steady state, we have to run for several times, then The average were taken and inputs changed again. Experiments started with 5 red points and 5 blue points and continues up to 50 points per color. Based on different outputs, the recommended algorithm score and the optimal algorithm score are derived from formulas 2 and 3:

$$\textbf{Recommended algorithm score} = \frac{(\text{blue cover} - \text{red cover})}{(\text{blue points})} * \mathbf{100} \tag{2}$$

$$\textbf{Optimal algorithm score} = \frac{(\text{blue optimal cover} - \text{red optimal cover})}{(\text{blue points})} * \mathbf{100} \tag{3}$$

Figure 8 shows the recommended and optimal algorithm scores based on the sum of red and blue points. In this chart from left to right the number of red points increases and rises from 5 to 50. If the number of blue points is steady in all 10 red states, and fluctuating starts from 5, up to 50 and drops back to 5 points (up to 50 points).

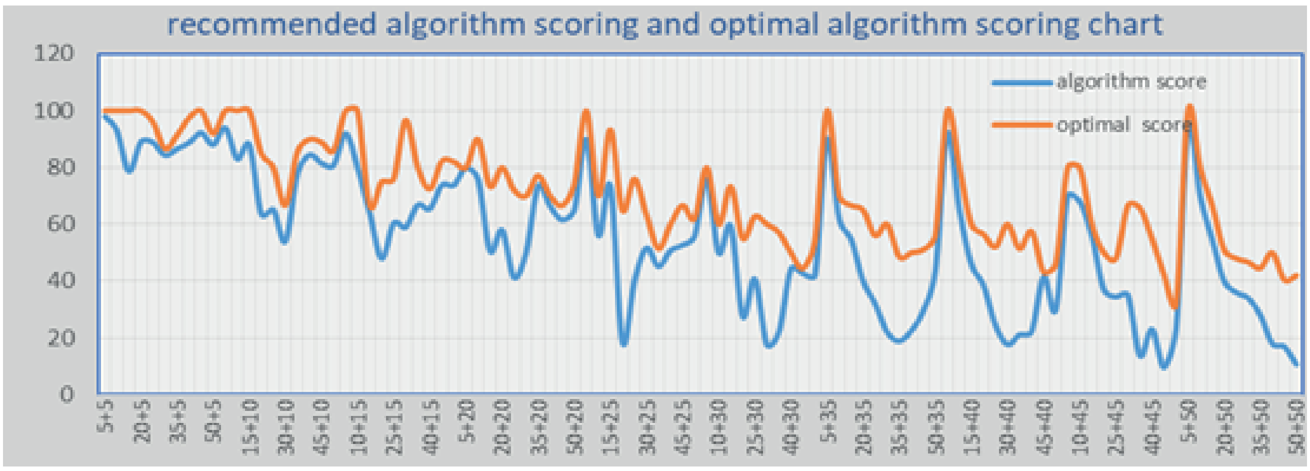


Figure 8: Algorithm Scoring charts

As shown in Figure 8, the optimal algorithm is higher than our algorithm; and in some cases is equal to recommended algorithm and overlapping with it. Also, the recommended algorithm in compare with optimal algorithm has a small difference in score, indicating its proper function. The blue and red covering chart is also shown in Figure 9. In this chart, the blue and red points are considered separately. According to the results, with the increase in the number of red points, their coverage is also increased.

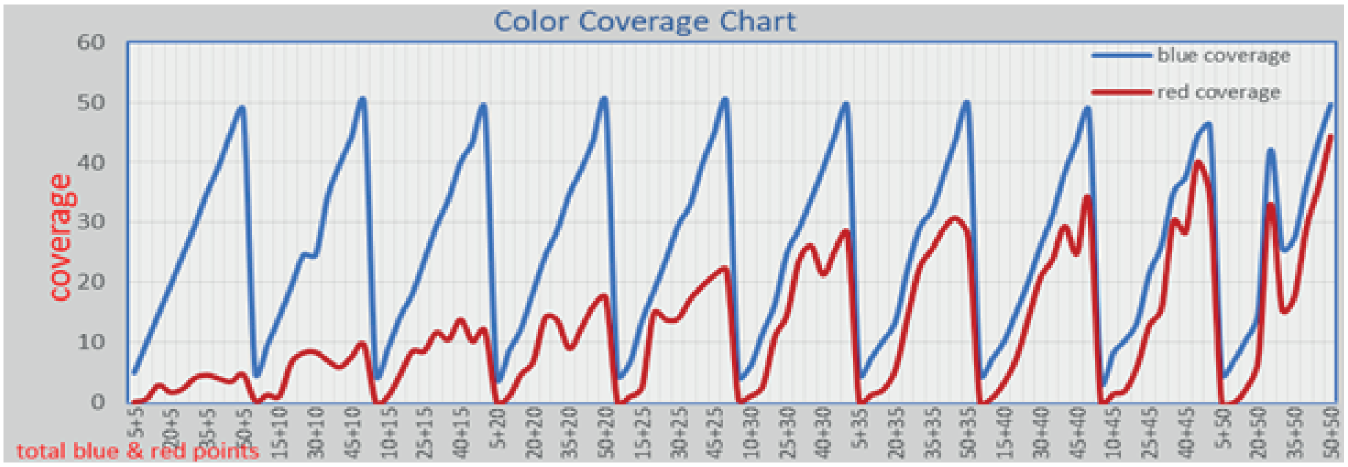


Figure 9: Color Coverage Chart

Figure 10 shows the optimal red points and optimum blue points. According to this chart, the number of red and blue coverage increases. At first, the difference between the blue and red charts is high and gradually decrease with increase in the number of red points; Also at the same time the red point's coverage will increase. In fact, with the increase in the number of red points, the algorithm's performance becomes more difficult in separating, and more number of red points will covered by rectangles.

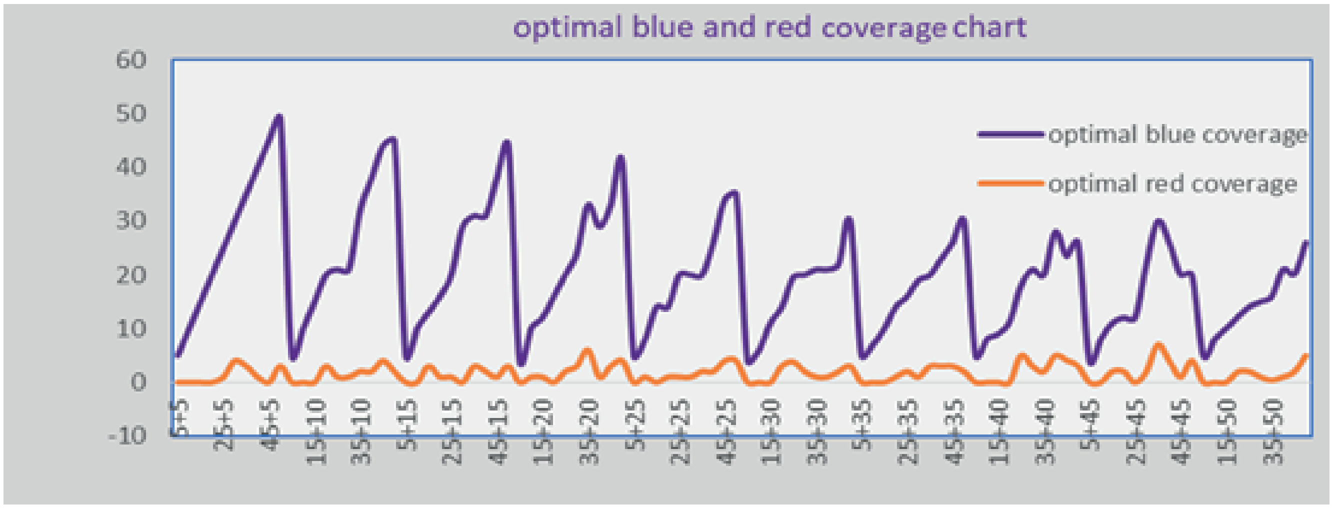


Figure 10: optimal coverage chart of blue and red points

The graphs and outputs derived from the average of the recommended algorithm score and optimal algorithm score (based on the sum of the points) are also shown in Figure 11. In this chart, the X axis represents the sum of the blue and red points and the Y axis represents the score. The blue curve is the average of recommended algorithm scores.

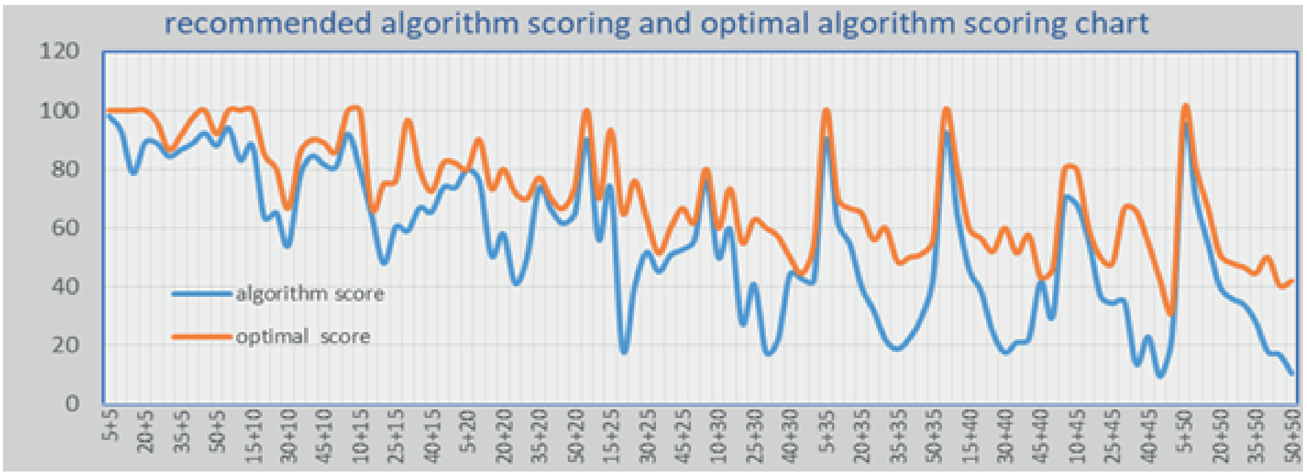


Figure 11: The recommended algorithm scoring and optimal algorithm scoring chart

As shown in Fig 12, our algorithm is always below optimal, but close to it, and does not have much difference to be optimal. In this chart, when the number of red and blue points increases, the coverage of red points also increases, and thus the score of the algorithm decreases.

As the total number of points increases, the algorithm score and optimal score gradually decreases and the algorithm separation becomes more difficult. But in general, the separation function of our algorithm is not far from the optimal algorithm, and by comparing the chart of these two algorithms, it can be concluded that our recommended algorithm in separation, has shown an acceptable result.

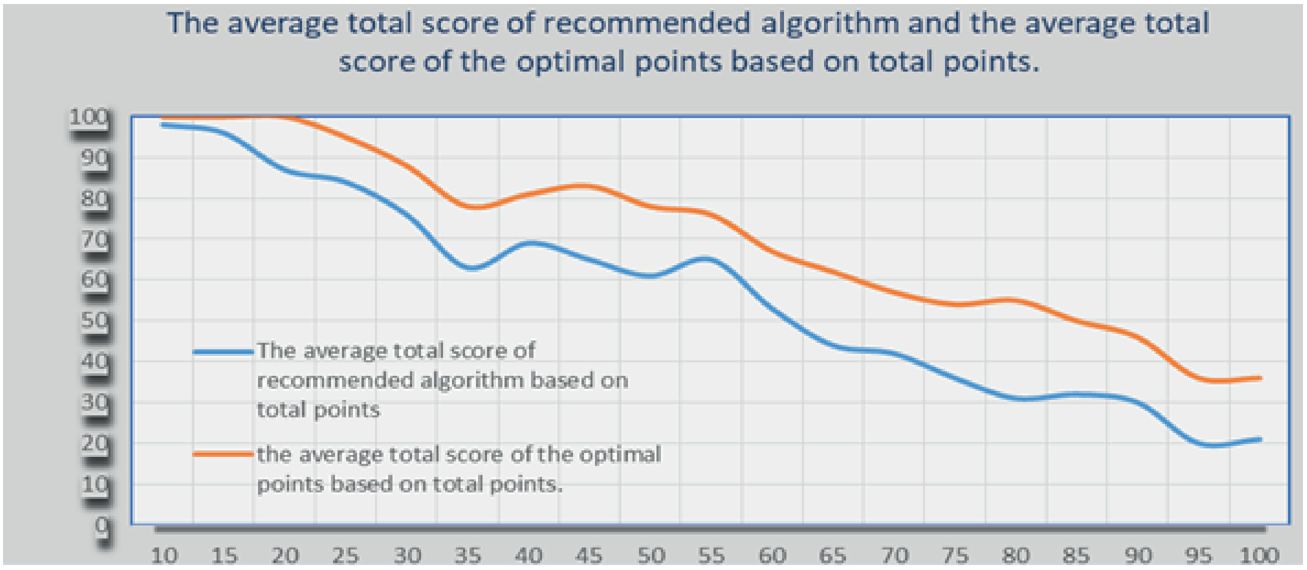


Figure 12: The average total score of recommended algorithm and the average total score of the optimal points based on total points.

To evaluate the performance of the recommended algorithm, with respect to the optimal algorithm, the ratio of the algorithm's score to the optimal algorithm score has been calculated. To obtain this ratio, the average optimal score is divided by the average score of the algorithm. As shown in Table 4, the recommended algorithm is operating as 0.8 of optimal algorithm, and in fact it has 0.2 distance with it.

Table 4. Final Scores

algorithm	score
optimal algorithm	70
recommended algorithm	56
Ratio of the recommended algorithm to the optimal algorithm	0.8

In order to Evaluate the efficiency of the recommended algorithm and the optimal algorithm the program runs about 1000 times and tests with different points and different input values.

Based on the data from the execution, it can be concluded that our proposed simulated annealing algorithm is accurate and efficient to solve the problem of separating the set of two-color points in one page.

7. CONCLUSION

In this paper, the problem of separation of two red and blue points is investigated by three rectangles and a new metaheuristic algorithm based on simulated annealing is presented. In general, simulated annealing algorithm has a good efficiency and precision, to solve the problem of separating two-color points on one page; and in many cases it provides an optimal or close response. In fact, our goal was to separate the desired blue points from undesirable points in red by three rectangles, in such a way that these rectangles contain the most desire points.

The recommended algorithm to solve this optimization problem, has been able to separate the blue points from the **n** input point, at time order (O) n and gain %80 of the optimal algorithm score. It also solve the limitations that exist in separation of points by one rectangle and two rectangles. On the other hand, our recommended method is provided, regardless of limitations or specific state, and a wide range of problems are solved by it.

In order to analyze the recommended method, the algorithm is executed with c# and is compared with the results of the optimal algorithm. Our results showed that this heuristic algorithm based on simulated annealing algorithm is near optimal, and in some cases it obtains the exact optimal response.

REFERENCES

1. Hanabadi Farahani, Afsaneh. (2013), "providing algorithms for separating color points with some geometric shapes," Master thesis, Faculty of Computer engineering, North Tehran University, Tehran.
2. Moslehi, Zahra. (2012), "Separability of points with geometric objects in two dimensional space," Master thesis, Faculty of Computer Engineering and Information Technology, Amir Kabir University, Tehran.
3. Khalafi, Sarah (2013), "An algorithm for the separation of points within polygonal environments using the minimum number of distinctive chord" Master thesis, Faculty of computer engineering, North Tehran University, Tehran.
4. Sheikhi, F. (2010), "Covering points in the Page Using Geometric Shapes," Master thesis, Faculty of Computer Engineering and Information Technology, Amir Kabir University, Tehran.
5. Mitsunori Miki, Satoru Hiwat, Tomoyuki Hiroyasu,(2006), "Simulated Annealing using an Adaptive Search Vector", 1-4244-0023-2006 IEEE.
6. Sylvester J.J., (2008), "A question in the geometry of situation", Quart. J. of Math. 1 (1857) 79.
7. Segal M. and Kedem K., (1998), "Enclosing k points in the smallest axis parallel rectangle", , Information Processing Letters, vol. 65, no. 2, pp. 95-99. https://doi.org/10.1016/S0020-0190(97)00212-3
8. Seara C., (2002), "Geometric separability", in Applied Mathematics, Ph.D. Thesis, Universitat Polit'ecnica de Cataluny.

9. Armaselu B. and Daescu O., (2016), "Dynamic minimum bichromatic separating circle", Theoretical Computer Science. https://doi.org/10.1007/978-3-319-26626-8_50
10. De Pano N.A.A., (1987), "Rotating calipers revisited: optimal polygonal enclosures in optimal time", ACM South Central Reginal Conf., Lafayette LA.
11. Freeman H. and Shapria R., (1975), "Determining the minimum area enclosing rectangle for an arbitrary closed curve", Commun. of the ACM 18, pp. 409-413. https://doi.org/10.1145/360881.360919
12. Eckstein J., Hammer P . L., Liu Y ., Nediak M. and Simeone B., (1992), "Finding minimum area kgons", Discrete & Computational Geometry (DCG), vol. 7, pp. 45-58. https://doi.org/10.1007/BF02187823
13. Khalafi Sara, Bagheri Alireza, Riahi Kashani M. M., (2014), "Finding The Biggest Monochromatic Polygon", INTERNATIONAL JOURNAL OF Scientific & Technology Research Volume 3, Issue 3.
14. Moslehi Zahra, Bagheri Alireza, (2016), "Separating bichromatic point sets by two disjoint isothetic rectangles", Scientia Iranica D. https://doi.org/10.24200/sci.2016.3891