

SIMULATED ANNEALING APPROACH FOR UNIVERSITY TIMETABLE PROBLEM

R. Kristoforus J. Bendi¹, Hadi Junaidi²

Universitas Katolik Musi Charitas^{1,2}

Bangau Street No. 60 Palembang, South Sumatera, Indonesia 30113

Su-rel: kristojb@ukmc.ac.id¹

Abstract : University Scheduling is a way of allocating students, lecturers, and rooms, which is used for lectures in the available time slots. The common problems are that a lecturer is scheduled in the same time slot, or several courses are scheduled in the same room and the same time slot. For this reason, scheduling needs to be made in such a way that it can optimize the use of resources. We use simulated annealing as an approach to solve that problem. The results showed that the higher the initial temperature value used and the greater the iteration value would reduce the violation constraints in scheduling problems.

Keywords: university timetable problem, simulated annealing, optimization.

Abstrak : Penjadwalan di Universitas merupakan cara untuk mengalokasikan waktu bagi mahasiswa, dosen dan ruangan yang digunakan untuk kegiatan perkuliahan. Permasalahan umum yang sering ditemui adalah bahwa seorang dosen dijadwalkan pada waktu yang sama atau beberapa mata kuliah dijadwalkan diruang yang sama dan pada waktu yang sama. Berdasarkan kondisi tersebut, perlu dibuat sistem penjadwalan sedemikian rupa sehingga dapat mengoptimalkan penggunaan sumber daya yang ada. Dalam penelitian ini, digunakan pendekatan dengan metode Simulated Annealing untuk memecahkan masalah tersebut. Hasil Penelitian menunjukkan bahwa semakin tinggi nilai pengukuran awal yang digunakan dan nilai iterasi yang lebih besar akan dapat mengurangi kendala dalam masalah penjadwalan.

Kata Kunci: Masalah Penjadwalan , simulated annealing, optimasi.

1. INTRODUCTION

Scheduling allows various activities to be carried out regularly and does not interfere with each other. According to [1] scheduling is an activity carried out to allocate existing resources or machines to run a set of tasks within a certain period of time. Research on scheduling has been widely discussed in various studies, including University Timetable Problem (UTP) [2,3,4,5,6].

Preparation the schedules is part of routine learning activities were done every semester at a college. Each college manages several departments with a number of students. The process of learning activities requires the

availability of classrooms, lecturers, and courses.

Based on observation, we have found that classroom utilization is less than 70%. It means there is a possibility to be optimized. In addition, the number of lecturers available is limited, and the teaching time preferences of lecturers are very diverse because they have to carry out various other activities besides learning activities. It causes limited time allocation from lecturers for learning activities.

Lecture scheduling is prepared by each department, after obtaining information on the lecturers' time preferences to be scheduled. After that, the schedules of each department will be combined. The problems that can arise in this

process are: (1) It often happens that the same room is scheduled for two or more different subjects at the same time, (2) sometimes a lecturer who has been scheduled in a department is also scheduled for other departments at the same time, (3) the students cannot take up to the maximum number of credits because courses that can be taken are scheduled at the same time, so that only one subject can be taken, and (4) we found that courses in the same semester in a department are scheduled at the same time. This causes students in the semester to only be able to take one of the courses.

The problems mentioned above were found when the class schedules of each department were combined, and some were even discovered during the period of filling out the study plan by the students. These problems cause the schedule repair process to be very difficult to do, because changing the schedule of one course will affect the overall schedule.

2. EXPERIMENTAL SECTION

2.1 University Timetable Problem

University Timetable Problem (UTP) is a classic problem in educational institutions that has attracted many researchers in the past two decades [2]. Scheduling in higher education can be divided into scheduling lectures and scheduling examinations [7]. There are many models that have been proposed in various studies. Jat & Yang [8] mentions UTP as a multidimensional allocation problem, where students and lecturers are allocated in courses, subject classes and time preferences (both

lecturers and students) are allocated in classrooms and timeslots. UTP is a type of time allocation problem that is resolved by evaluating the limitations given. Every educational institution has different characteristics. This causes the boundaries made to be different for each institution. In general, the limitations often encountered in lecture scheduling problems are as follows [9]: (1) edge constraint, two events may not occupy the same time slot, (2) ordering constraints, limits that govern the sequence of events, (3) event spread constraints, limits that govern the spread of events in a scheduling, (4) preset specification and exclusion, a limit that determines in advance the time slot that will be used by an event before the solution search process, and (5) capacity constraints, restrictions that are related to room capacity

Kohshori & Abadeh [10] grouped boundaries into hard constraints and soft constraints. Hard constraints are the limits that should not be violated at all. While the soft constraint is a limitation that as much as possible to be fulfilled.

2.2 Simulated Annealing

Simulated Annealing (SA) was introduced in a paper published by Metropolis in 1953. If we heat a hard material until it melts and then cool it, then the nature of the material structure depends on the level of cooling. If the liquid material is cooled slowly, it will produce good quality of crystals. Conversely, if liquid material is cooled rapidly, the formed crystals will not be perfect. The algorithm proposed by Metropolis simulates matter as a system of particles. The

algorithm simulates a cooling process that gradually lowers the temperature of the system until it converges to a frozen and stable state.

Kirkpatrick [11] used the idea of the Metropolis algorithm and applied it to optimization problems. The idea is how to use SA to find feasible and converging solutions to optimal solutions. They implemented SA in the optimal design of computer hardware and the Traveling Salesman Problem (TSP).

Annealing is a cooling of metal slowly, after the metal has been heated at a very high temperature. The cooling process of metal heated at high temperatures takes place slowly when the temperature drop stops, the metal has been in a condition with very low energy. The SA algorithm simulates the annealing process in making material consisting of glassy or metal grains. The purpose of this process is to produce a good crystal structure using minimum energy. When resolving the optimal problem using SA, the structure of a substance will represent the preparation of the solution of a problem and the temperature is used to determine how and when new solutions can be updated and accepted. This algorithm is basically a three-step process, namely updating the solution, evaluating the quality of the solution and determining the solution received.

Implementing SA requires a number of random numbers. Choosing the right random generator requires special knowledge of a problem. Basically it is important to specify the number of random numbers needed and determine the speed of the generator. The random generator will produce the initial

solution. In addition to using random builders, the formation of solutions can also be done with other algorithms such as search algorithms so that the SA algorithm speed will be better.

```

Find a random initial solution  $s := s_0$  using  $cs(\mathcal{P})$ 
Select an initial temperature  $t := t_0 > 0$ 
Select a temperature reduction function  $\alpha$ 
repeat
  repeat
     $s' := NeighborhoodSearching(s)$ 
     $\delta := F(s') - F(s)$ 
    if( $\delta \leq 0$ ) or ( $\exp(-\delta/t) < \text{rand}[0, 1]$ )
       $s := s'$ 
    endif
  until iteration_count =  $n_{rep}$ 
   $t := CoolingSchedule(t)$ 
until stopping condition is true

```

Figure 1. General Simulated Annealing Algorithm [12]

The SA algorithm (Figure 1) accepts a new solution if the new cost is lower than the cost of the current solution in each iteration. With these criteria, SA allows avoiding local traps (local minimum). This is one of the advantages of SA compared to other scheduling methods.

Simulated Annealing has three parameters: initial temperature, reduction value, and number of iterations. These parameters are important parameters in the successful implementation of the SA algorithm [13]. The functions of the three parameters are:

1. Initial Temperature; used to determine the length of the process of making a new solution, because the higher the number determined and the smaller the value of the temperature reduction used to decrease, the longer the temperature decreases. Determination of the initial temperature value is adjusted to the problem at hand, in determining the initial temperature precisely

until now there is no single method that can accurately determine the initial temperature for all problems. So this value can be changed depending on when testing the algorithm for the problem. The end point of temperature drop is reaching zero temperature.

2. Value of Reduction; is a fixed value that will be used to reduce the initial temperature. According to [14], this value is determined based on experience which shows that α that should be used is between 0.8 and 0.99. Usually a good result is obtained when α approaches 0.99. The greater the value of α the longer the process of decreasing the temperature to reach the stop criteria.
3. Number of iterations; if the amount of iteration is too large it will help achieve thermal balance at each temperature but with increased computational time. If the number of iterations is small, it can cause an increase in the speed of convergence or lead to a solution but the optimum local.

Various studies related to the use of SA in UTP have been done previously. Rochman [15] in his research developed a course scheduling application using the Simulated Annealing algorithm at Trisakti University. The results showed that the execution time needed to form this initial schedule was relatively fast at 488 milliseconds. Another research [16] combined vertex graph coloring (VGC) and simulated annealing in their research. Their experiments show that the combination of VGC and SA can avoid scheduling problems. The test results show

that 100% of the problems of lecture and lecturer clashes can be avoided, and 93.89% clash of room use can be avoided. The fulfillment of hard constraints is 98.47%, and the fulfillment of soft constraints is 82.4%. Overall the system success rate is 90.435%. Sari and Suseno [17] conducted a study on the optimization of college scheduling using the simulated annealing method. This study aims to make the scheduling of courses in college using the simulated annealing method with five data variables, namely lecturers, subjects, time slots consisting of days and time periods and space variables. Validation is done by testing the simulated annealing method by producing a variant average of 77.791% of data can reach a solution with a standard deviation of 3.931509. In this study a solution method was used to use the remaining search space to be reused by unallocated data.

2.3 Methods

This research activity consists of the following steps: (1) problem analysis, (2) literature study, (3) data collection, (4) application design (5) simulation of SA, and (6) testing and analysis. The scheduling problems are based on the Faculty of Science and Technology, Universitas Katolik Musi Charitas. We used data from the 2016/2017 academic year. The software was modeling and designing by Unified Modeling Language (UML). We developed the software with Visual Basic 2015 and mySQL. The data is inputted to the software to be simulated. The simulation results will be discussed.

3. RESULT AND DISCUSSION

3.1 Algorithm Design

Based on [12], the scheduling steps with SA can be defined as follows:

1. Determine the initial value for each of the following parameters: temperature (T), temperature reduction factor (α), and number of iterations (I).
2. Set $k = 0$.
3. Generate the initial schedule ($S = S_k$). As a note, in our scheduling software, the initial schedule were generate with graph coloring method (Sunarni et al, 2017; Sunarni et al, 2018).
4. For each element in S_k , if there is a violation of the constraint, give a value of 1 and vice versa give a value of 0.
5. Calculate the number of violations (V_k) in S_k by summing all the values of the slot elements.
6. Set $k = k + 1$.
7. Calculate the value of R_1 with the formula: $R_1 = 1 + \text{INT}(2r_1)$, with r_1 = random numbers that have a uniform distribution in $[0,1]$ and INT means only the integer part is taken.
8. If $R_1 = 1$, it means that the row (session) will be shifted. If $R_1 = 2$, it means that the column (days) will be shifted.
9. Calculate R_2 and R_3 with the formula $R_2 = 1 + \text{INT}(Nr_2)$ and $R_3 = 1 + \text{INT}(Nr_3)$, with r_2 and r_3 = random numbers that have uniform distributions in $[0,1]$, N is the number of rows or column in S (depending on the results of step 8), and INT shows means only the integer part is taken.
10. Swap rows or columns R_2 with rows or columns R_3 .
11. Arrange the new schedule (S_k)
12. Calculate the number of violations (V_k) by summing all the values of the slot elements.

13. If $V_k \leq V_{k-1}$, then do the following steps: (a) set $S = S_k$, (b) Go to step 15.
14. If $V_k > V_{k-1}$, then do the following steps: (a) Generate a random value of x , (b) Calculate the probability value of $P = e^{(-\Delta V/T)}$, with $\Delta V = V_k - V_{k-1}$. (c) If $P \geq x$ then set $S = S_k$ go to step 15
15. Set $T_k = \alpha T_{k-1}$
16. If $k < I$ or $T > 0$ returns to step 4. If not, stop the process.

3.2 An Illustration

For example, the following will be given an illustration. The first step in SA is to determine the initial value for each parameter. Suppose the initial value for temperature (T) = 500, the temperature reduction factor (α) = 0.99, and the number of iterations (I) = 3. The next step is to generate the initial schedule (S_0). Suppose S_0 as in Table 1.

The next step is to check all slots that have constraint violations. For example, *Course 1* can only be held in *Session 2*, and *Course 7* can only be held on *Day 2*. The slot elements that have any constraint violation will be given a value of 1, and slot elements that do not have any constraint violations will be given a value of 0. Thus the data in the Table 1 can be rewritten as Table 2. To calculate the number of constraint violations that occur, we add the values of all the elements in the slot, so that the number of violations obtained is: $V_0 = 1 + 0 + 0 + 0 + 0 + 0 + 1 + 0 = 2$.

Table 1. The initial schedule

	Day 1	Day 2	Day 3	Day 4
Session 1	Course 1	Course 2	Course 3	Course 4
Session 2	Course 5	Course 6	Course 7	Course 8

Table 2. The violation constraints of S

	<i>Day 1</i>	<i>Day 2</i>	<i>Day 3</i>	<i>Day 4</i>
Session 1	1	0	0	0
Session 2	0	0	1	0

Then, when $k = 1$, we calculate values of R_1 , R_2 , and R_3 . Suppose the random value $r_1 = 0.95$, $r_2 = 0.43$, and $r_3 = 0.68$ (see Table 3). Then the value of $R_1 = 1 + \text{INT}(2(0.95)) = 2$. Because of $R_1 = 2$, the swapping process will be done in the column. In this case $N = 4$ (number of columns = 4). Next we calculate the values of R_2 and R_3 . $R_2 = 1 + \text{INT}(4(0.43)) = 2$, and $R_3 = 1 + \text{INT}(4(0.68)) = 3$. It means that the columns will be shifted are the second and third columns in S .

Table 3. The values of variable for each k

k	r₁	R₁	N	r₂	R₂	r₃	R₃	x	P	S	V	T
0	-	-	-	-	-	-	-	-	-	S_0	2	500
1	0.95	2	4	0.43	2	0.68	3	-	-	S_1	1	495
2	0.21	1	2	0.39	1	0.57	2	-	-	S_2	0	490
3	0.82	2	4	0.08	1	0.31	2	0.54	0.99	S_3	1	485

The next step is to reorder the schedule after the column exchange is done. The schedule is obtained as in Table 4. Next we recalculate the number of violation constraints (V_k). Based on Table 5, it is obtained $V_k = 1$. Because $V_1 < V_0$, the new schedule in Table 4 is used as the initial schedule (S_1) for the next iteration. Furthermore, the value of T_k is calculated by $T_k = \alpha T_{k-1} = (0.99)(500) = 495$.

The calculation results for the next iteration can be seen in Table 3. On the second iteration, which is shifted is the first and the second row. The result is S_2 (Table 6). Based on table 7, it can be seen that $V_k = 0$. This means that no

violations occurred. But because it has not reached the number of iterations $I = 3$ or $T = 0$, the iteration will continue. The results obtained in the third iteration (Table 8) indicate that there is one constraint violation ($V_k = 1$). Because the value of $V_k > V_{k-1}$, the P value is calculated (Table 3). The results of the calculation of the P value indicate that the new schedule (S_3) is accepted as the result schedule.

Table 4. The schedule of S_1

	<i>Day 1</i>	<i>Day 2</i>	<i>Day 3</i>	<i>Day 4</i>
Session 1	Course 1	Course 3	Course 2	Course 4
Session 2	Course 5	Course 7	Course 6	Course 8

Table 5. Violation of constraint in S_1

	<i>Day 1</i>	<i>Day 2</i>	<i>Day 3</i>	<i>Day 4</i>
Session 1	1	0	0	0
Session 2	0	0	0	0

Table 6. The schedules of S_2

	<i>Day 1</i>	<i>Day 2</i>	<i>Day 3</i>	<i>Day 4</i>
Session 1	Course 5	Course 7	Course 6	Course 8
Session 2	Course 1	Course 3	Course 2	Course 4

3.3 Software Design

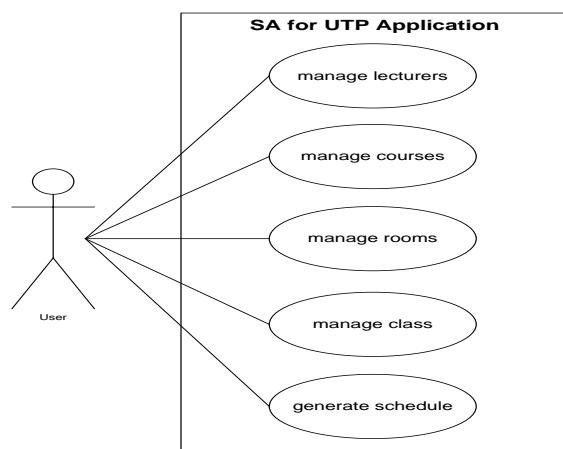
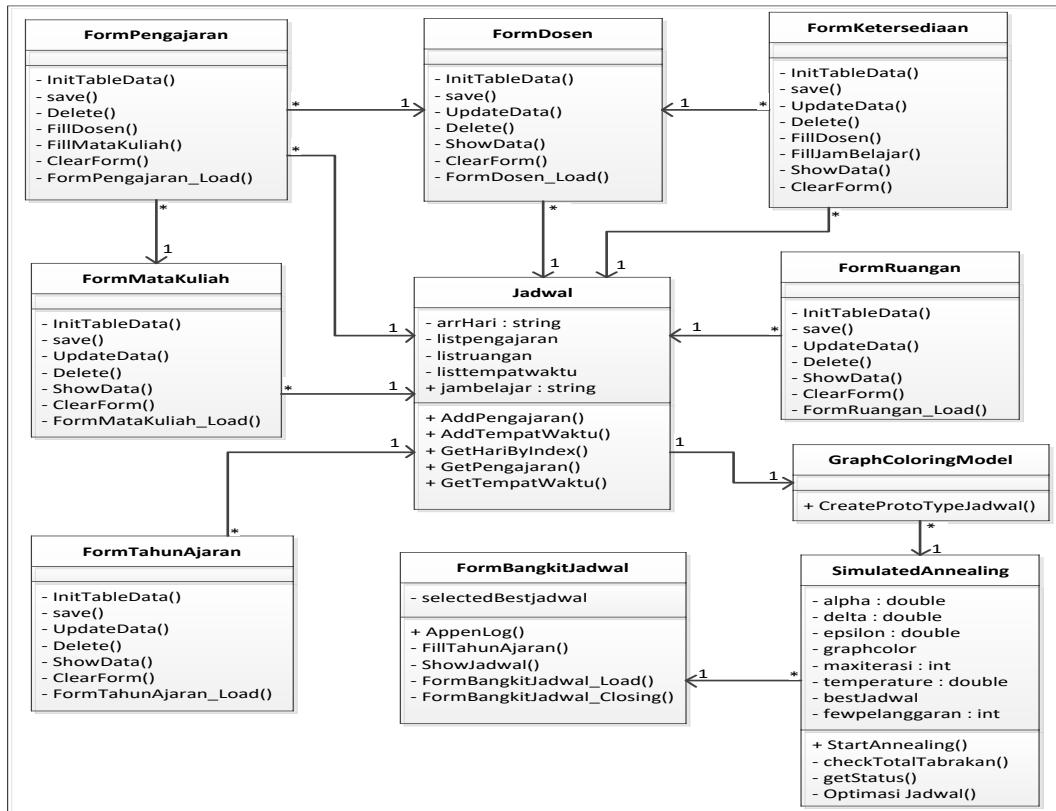
Modeling software is a method used to describe the basis of the system to be built. Software models were developed with the Unified Modeling Language (UML). We used four diagrams to model the software: (1) use case diagram is used to describe system and user interactions (Figure 2), (2) class diagrams to model the static structure of the system (Figure 3), (3) sequence diagrams to model an interaction on the system, (4) activity diagrams to model the behavior of a system.

Table 7. Violation of constraint in S₂

	<i>Day 1</i>	<i>Day 2</i>	<i>Day 3</i>	<i>Day 4</i>
Session 1	0	0	0	0
Session 2	0	0	0	0

Table 8. The schedules of S₃

	<i>Day 1</i>	<i>Day 2</i>	<i>Day 3</i>	<i>Day 4</i>
Session 1	Course 7	Course 5	Course 6	Course 8
Session 2	Course 3	Course 1	Course 2	Course 4

**Figure 2. Use case of scheduling software****Figure 3. Class diagram of scheduling software**

3.4 The Results

Testing is done with a number of data. Data collection is done by retrieving data in even semester 2017/2018. We conducted 69 experiments with initial temperatures ranging from 150 to 1300 and the number of iterations ranged from 50 to 475. We used a fixed alpha value of 0.99. Table 9 shows the average value of the whole experiment.

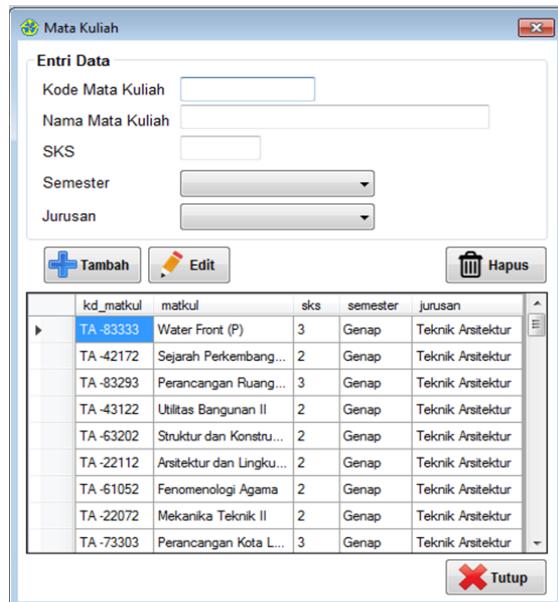


Figure 4. User interface for manage the courses

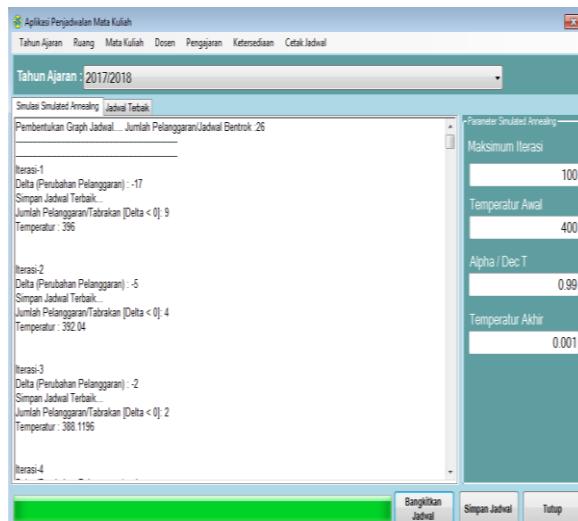


Figure 5. User interface for simulated annealing process

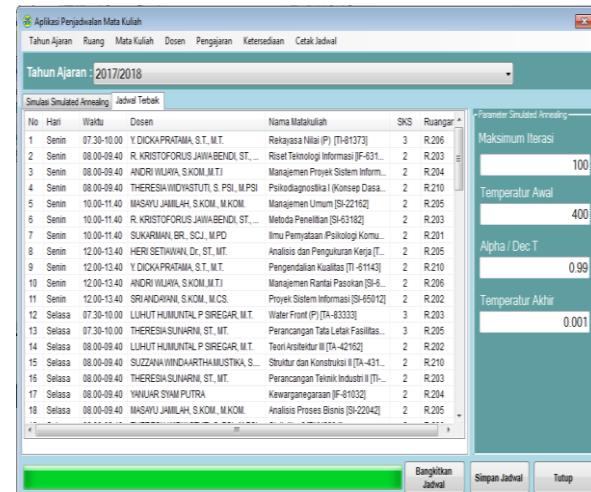


Figure 6. User interface for scheduling process result

Table 9. Teh experiment results

No. of iteration	T = 150	T = 300	T = 400	T = 500	T = 800	T = 1000	T = 1300	Average constraints violation
50	2.0	2.0						2.0
75	2.0	2.0	3.0					2.3
100	1.0	0.0	2.0	3.0	3.0	1.0	1.0	1.6
125	0.0	3.0						1.5
130		3.0	0.0	1.0				1.3
150	0.0	0.0	3.0	2.0	1.0	2.0	2.0	1.4
175		0.0			2.0			1.0
180			1.0	1.0				1.0
200	0.0	0.0	1.0	1.0	1.0	2.0		0.8
225	2.0	1.0	1.0					1.3
250	0.0	1.0	0.0	1.0	2.0	0.0		0.7
275	1.0	1.0	0.0	0.0				0.5
300		0.0	1.0	2.0	1.0	0.0		0.8
325			0.0	2.0				1.0
350				0.0	1.0	0.0		0.3
375					1.0	0.0		0.5
400					0.0	2.0		1.0
425						1.0		1.0
450					0.0	0.0		0.0
475					0.0			0.0
Average constraints violation	1.0	1.0	1.5	0.9	1.3	0.9	0.8	1.0

The results of the analysis (shown on Figure 7 and Figure 8) show that the higher the initial temperature value, the smaller the violation of the constraints that occur. Likewise, the more iterations will reduce the number of violation constraints.

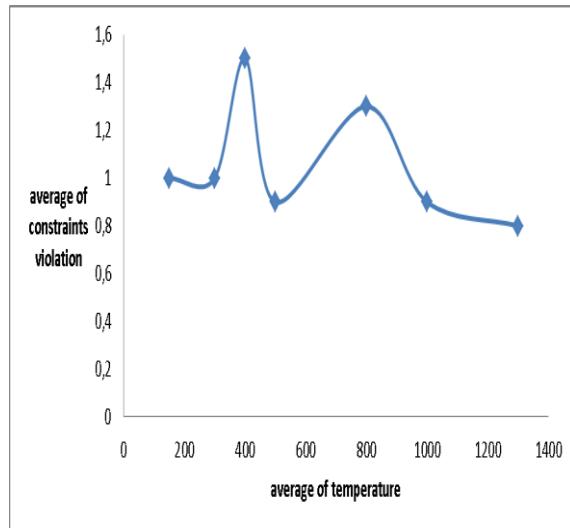


Figure 7. Correlation between temperature and constraints violation

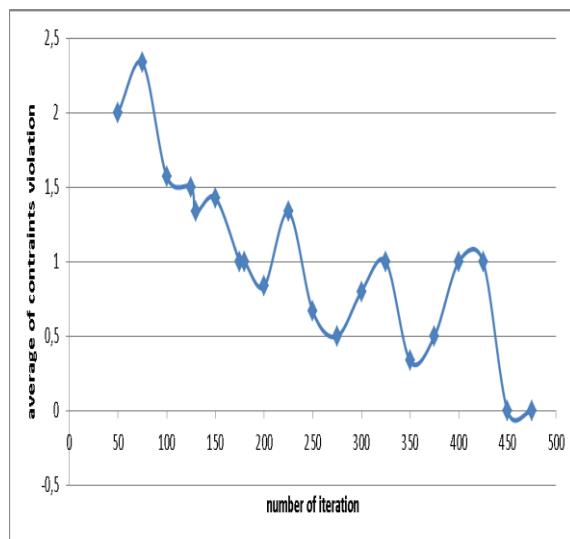


Figure 8. Correlation between number of iteration and constraints violation

4. CONCLUSION

The results show that SA can be used as an approach to solve a university timetable problem. The higher the initial temperature and the more the number of iterations can reduce the number of violation constraints. Further research can be done to find the ideal value of the initial temperature and the best minimum number of iterations.

ACKNOWLEDGEMENT

This research is part of National Strategic Research (Penelitian Strategis Nasional) funding by the Ministry of Research, Technology and Higher Education, Indonesia. With the number of contract is 2182/SP2H/LT/K2/KM/2018. We would like to thank all those who contributed to this research, especially our colleagues at Universitas Katolik Musi Charitas.

REFERENCES

- [1] K.R. Baker, *Introduction to Sequencing Scheduling*.: John Wiley & Sons, 1974.
- [2] Z. Bozyer, M. S. Basar, and A. Aytekin, "A Novel Approach of Graph Coloring for Solving University Course Timetabling Problem," in *The Second International Symposium on Computing in Science & Engineering*, Kusadasi, Aydin, Turkey, 2011, pp. 560-566.
- [3] F. K. Dewi, "Pembangunan Perangkat Lunak Pembangkit Jadwal Kuliah dan Ujian Dengan Metode Pewarnaan Graf," *Buana Informatika*, vol. 1, no. 1, pp. 57-68, 2010.
- [4] M. Mariana and L. Hiryanto, "Penjadwalan Kelas Matakuliah menggunakan Vertex Graph Coloring dan Simulated Annealing," *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 1, no. 1, pp. 125-132, 2013.
- [5] W. A. Puspaningrum, A. Djunaidy, and R. A. Vinarti, "Penjadwalan Mata Kuliah Menggunakan Algoritma Genetik di Jurusan Sistem Informasi ITS," *Jurnal Teknik POMITS*, vol. 2, no. 1, pp. 127-131, 2013.
- [6] N.R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A Honey Bee Mating Optimization Algorithm For Educational Timetabling Problems," *European Journal of Operational Research*, vol. 216, no. 2012, pp. 533-543, 2012.
- [7] K. Setemen, "Implementasi Algoritma Genetika Dalam Pengembangan Sistem

- Aplikasi Penjadwalan Kuliah," *Jurnal IKA*, vol. 8, no. 1, pp. 56-68, 2010.
- [8] N. S. Jat and S. Yang, "A Memetic Algorithm for the University Course Timetabling Problem," in *IEEE International Conference on Tools With Artificial Intelligence*, Dayton, OH, USA, 2008, pp. 427-433.
- [9] H. Saragih, G. Hoendarto, B. Reza, and D. Setiyadi, "Aplikasi Sistem Perangkat Lunak Menggunakan Algoritma Ant Untuk Mengatur Pendjadwalan Kuliah," *Jurnal Teknik dan Ilmu Komputer*, vol. 1, no. 3, pp. 241-256, 2012.
- [10] M. S. Kohshori and M. S. Abadeh, "Hybrid Genetic Algorithms for University Course Timetabling," *International Journal of Computer Science Issues*, vol. 9, no. 2, pp. 446-455, 2012.
- [11] S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.
- [12] E. Aycan and T. Ayav, "Solving the course scheduling problem using simulated annealing," in *IEEE International Advance Computing Conference*, Patiala, India, 2009, pp. 462-466.
- [13] B. Santosa and T. J. Ai, *Pengantar Metaheuristik Implementasi dengan Matlab*. Surabaya: ITS Tekno Sains, 2017.
- [14] Suyanto, *Algoritma Optimasi Deterministik atau Probabilistik*. Yogyakarta: Graha Ilmu, 2010.
- [15] A. Rochman, "Penjadwalan Kuliah Menggunakan Metode Constraints Programming dan Simulated Annealing," in *Seminar Nasional Aplikasi Teknologi Informasi*, Yogyakarta, 2012.
- [16] T. Kristanto, T. Indriyani, and N. Khoiroh, "Penjadwalan Ruang Kuliah Menggunakan Vertex Graph Coloring Dan Simulated Annealing," in *Seminar Nasional Sains dan Teknologi Terapan*, Surabaya, 2016, pp. 61-68.
- [17] W. Sari and J. E. Suseno, "Metode Simulated Annealing untuk Optimasi Penjadwalan Perkuliahan Perguruan Tinggi," *Jurnal Sistem Informasi Bisnis*, vol. 6, no. 2, pp. 133-143, 2017.