Application of Viral Systems for Single-Machine Total Weighted Tardiness Problem

Budi Santosa and Umar Affandi
Industrial Engineering, Institute Technology of Sepuluh Nopember (ITS), Kampus ITS, Surabaya 60111, Indonesia

E-mail: budi_s@ie.its.ac.id, umar12345678910@gmail.com

Abstract. In this paper, a relatively new algorithm inspired by the viral replication system called Viral Systems is used to solve the Single-Machine Total Weighted Tardiness (SMTWTP). SMTWTP is a job scheduling problem which is one of classical combinatorial problems known as np-hard problems. This algorithm makes the process of finding solutions through neighborhood and mutation mechanism. The experiment was conducted to evaluate its performance. There are seven parameters which are required to tune in to find best solution. The experiment was implemented on data sets of 40 jobs, 50 jobs, and 100 jobs. The results show that the algorithm can solve 235 optimally out of 275 problems.

Keywords: Viral Systems, Single-Machine Total Weighted Tardiness Problem, Metaheuristics, combinatorial problem

1. Introduction
Scheduling and sequencing is one form in decision making that has an important role in the manufacturing and services industries. In the existing competitive environment, sequencing and scheduling becomes an important issue to maintain market share. Companies must determine the exact shipping time which is already determined based on agreement with the consumer. The failure of the fulfillment of this could lead to the loss of loyalty. Therefore, they need to schedule an activity based on the available resources in an efficient manner. [2].

Machine Scheduling Problems (MSPs) is one of the classical combinatorial optimisation problems [3]. Total weighted tardiness scheduling problem is one of MSPs that is not only a NP-hard problem [4], but also difficult to solve practically. Potts and Van Wassenhove [5] have found problem in solving problem of 50 jobs with state of the art branch and bound algorithm [6]. According to the difficulty of this problem, metaheuristics method are recommended to find a good solution for computing time reason. Metaheuristics have been successfully applied to many problems including Genetic Algorithm (GA), Tabu Search (TS), and Ant Colony Optimization (ACO) [7]. In this study, Viral Systems is proposed to solve Single Machine Total Weighted Total Tardiness Problem (SMTWTP).

Viral system is an optimization algorithm based metaheuristics introduced by Cortes et. al. [8]. This algorithm is relatively a new method that is inspired by analogy of virus behavior. The virus continues to reproduce by means of infectious cells. Then, the cells also strive to provide resistance in the form of antibodies. The paper is organized as follows. Section 2 explains about Viral Systems in brief.
Next, section 3 discusses the SMTWTP and section 4 elaborates the implementation of Viral Systems on SMTWTP. The numerical experiment is presented in section 5 and the results are concluded in Section 6.

2. Viral Systems

A transition creature between living things and inanimate objects named viruses are unique in terms of reproduction. The virus uses the cells of organisms as a container for replication. When the virus injects DNA into the host cell in the body, there are two types of replication: lytic cycle and lysogenic cycle. Lytic cycle allows the virus to replicate in the host cell. At a certain amount of viral replication will break the wall and out of the host cell. While the lysogenic cycle allows the virus to remain in the host cell is switched off until there is something to activate it. Then the cell of organism that has been injected was broken or mutated, it depends on the health of the cell itself.

In the system of the virus, the virus always seek and infect susceptible cells. Each cell was trying to produce antibodies to fight the viral infection. The cell that fail to produce antibodies will be grouped into the clinical picture. VS define the clinical picture of the infected population as a description of all the cells infected by viruses. In terms of computing, Clinical Picture containing solutions are being explored (the genome of infected cells, in biological terms) and total nucleus-capsids were replicated in the form of NR (for lytic replication) or IT (for lysogenic replication) [9].

In the infected cells, there are virus replication in the lytic or lysogenic. Both replication is done based on probability of lytic \( p_{lt} \) and the probability of lysogenic \( p_{lg} \) and \( p_{lt} + p_{lg} = 1 \). Initially, the number of nucleus capsid is zero (NR = 0 and IT = 0).

\[ \text{Figure 1 Clinical picture (Source: [8])} \]

2.1. Lisogenic replication

In the lysogenic replication, the activation process of mutation occurs after iteration reaches the maximum limit (LIT). LIT value depends on the health condition of the cells, so the cells are healthy (minimization function \( f(x) \) of high value) would have a lower probability of infection so the value of LIT becomes higher. In contrast, unhealthy cells will have a lower value of LIT [9]. The value of LIT can be calculated from the value of initial LIT (LIT\(^{0}\)) based objective function value relative to the best objective function has been found so far, which can be calculated by the following equation.

\[ \text{LIT}_{sel-x} = \text{LIT}^{0} \left( \frac{f(x)-f^{\theta}}{f^{\theta}} \right) \]

While:

- \( \text{LIT}_{sel-x} \) = LIT value of cell-x
- \( \text{LIT}^{0} \) = initial LIT value
- \( f(x) \) = Objective function value of cell-x
- \( f^{\theta} \) = The best value of objective function has been found so far.
2.2. Lytic replication

In lytic replication, first we need to obtain the number of replication nucleus capsid (NR). NR is calculated for each iteration as a function of the binomial variable (Z), its value is added to the existing NR on the clinical picture. Z is calculated using the binomial distribution is given by the maximum value of the nucleus replication capsid (LNR) and the probability of the single replication (p_r).

\[
z = \text{bin}(\text{LNR}, \ p_r) \quad \text{(2)}
\]

\[
\text{NR}_{\text{iter}} = \text{NR}_{\text{iter-1}} + z \quad \text{(3)}
\]

LNR represents the limit cell wall rupture and viruses that inhabit it out. LNR value depends on the value of the objective function (f(x)). Cells with the value f(x) which has a high probability of infection is lower, then the value will be higher LNR [9]. LNR Value can be calculated from the value of initial LNR (LNR^0) based objective function value relative to the best objective function has been found, which can be calculated by the following equation.

\[
\text{LNR}_{\text{sel-x}} = \text{LNR}^0 \left( \frac{f(x)}{f(x)} \right) \quad \text{(4)}
\]

where:

- LNR_{sel-x} : LNR value of cell-x
- LNR^0 : initial LNR value
- f(x) : Objective function value of cell-x
- f'(x) : The best value of objective function has been found so far.

After that, the cells that have a value exceeding NR LNR be broken so that the virus released. Each virus liberated would have probability (p_i) to infect new cells other being around. If the cardinality of x is defined as |V(x)|, the number of cells infected by the virus in the environment can be obtained by using the binomial random value |V(x)| and the probability p_i.

\[
y = \text{bin}(|V(x)|, p_i) \quad \text{(5)}
\]

On the other hand, to maintain from the growth of a viral infection, the organism (combined cell) responds by producing antigen. In the clinical picture, individual infected cells produce antibodies with Bernoulli probability distribution A(x) = Ber(p_{an}), which is the probability p_{an} antibodies released by cells in the clinical x-picture. Therefore, the population of infected cells produce antibodies based on a binomial distribution with parameters characteristic clinical picture size (n) and the probability of producing antibody (p_{an}) [9].

\[
A(\text{populasi}) = \text{Bin}(n, p_{an}) \quad \text{(6)}
\]

2.3. Mutation

Lysogenic replication allows the algorithms to perform searching by mutation. Mutation has a goal to bring new solutions that are totally different from the previous solution in order to get out of the local optimal solution [10]. A feasible solution can be mutated and transferred to obtain other feasible solutions. Given n jobs, each job is represented by a gene, there are several types of mutation, for examples [10]:

1. Flip

This mutation is done by reversing the mutated segment left-to-right to be right-to-left. Firstly, we choose two different random-integer uniform number at range 1-n, let them a and b, and a < b. Then we flip the segment between job a and b, [ja, ja+1, …, jb], reverse to be [jb, jb-1, …, ja].

2. Swap

This mutation is done by exchanging a gene with the other gene. Firstly, we choose two different random-integer uniform number at range 1-n, a and b, and a < b. Then, we interchange the position of a and b.

3. Slide

Firstly, we choose two different random-integer uniform number at range 1-n, a and b, and a < b. a is called the “head” of segment and b is called the “tail”. Then, place the head of backward and the other
genes of mutated segment are stepped forward, so the original segment [j_a, j_a+1, ... j_b] will become [j_a+1, j_a+2, ... j_b, j_a].

2.4. Neighborhood
The lytic replication allows the algorithms to perform searching by neighborhood. Neighborhood tracking is a method of solution adjacent to a good solution that has been found before. Therefore, this method makes the algorithm to do local search faster. Neighboring cells can be defined by exchanging a pair of genes in the genome of the cell before. Starting from the left gene first, interchange with a second gene that produced the first neighboring cells. For cells whose genome consists of n genes, the procedure is repeated until the (n-1) neighboring cells are obtained [11].
For example, the feasible solution 1-2-3-4-5 has following several neighborhood solutions:
The first job is exchanged with the second one. 2-1-3-4-5
The second job is exchanged with the third one. 1-3-2-4-5
The third job is exchanged with the fourth one. 1-2-4-3-5

3. Single machine total weighted tardiness problem
Single machine total weighted tardiness problem (SMTWTP) is known as a planning operation problem. In SMTWTP, a set of jobs j = (j_1, ..., j_n) need to be processed on a single machine. Each job (j_j) consists of only a single operation, which involves the processing time p_j > 0 \forall j = 1, ..., n. The penalty of tardiness, w_j > 0 \forall j = 1, ..., n. Machine can process only one job at a time, there was no parallel work processes. Each job j should be completed before the deadline d_j. If the job is done over time d_j, then the delay (lateness) is calculated as a function of the following:

T_j = max (s_j + p_j - d_j, 0) \tag{7}
The problem is also formulated to the following integer programming model [12]:

Minimize TWT = \sum_{j=1}^{n} (w_j, \max(0, s_j + p_j - d_j)) \tag{8}

Subject to: s_j - s_j' \geq p_j - M Z_{j,j'} \tag{9}

where
s_j = starting processing time of job j
p_j = processing time of job j
d_j = due date of job j
T_j = lateness of job j
w_j = weight of lateness of job j

with M is a large positive number and z_{j,j'} is a binary variable, equal to 1 if job j followed by job j', and 0 otherwise.

4. Viral Systems for SMTWTP
The following pseudocode describes the application of Viral Systems for SMTWTP.

Procedure VS_SMTWTP(N_{max}, CPS, p_{lt}, p_{l}, p_{an}, LNR, LIT, data) CP=∅
iteration = 0
for i = 1 to CPS
    /* Get randomly feasible solutions and their replication type
    CP(i) = Random permutation()
    CP(i).Replication_Type = Get_Randomly_Replication_Type (p_{lt})
    F(i) = SMTWTP (CP(i), data)
    Next
    Fbest = min (F)
Do
iteration = iteration+1
For c =1 to clinical_size
    If Replication_Type (CP (c)) = ‘Lytic’ Then
        Lytic (c , CP, \( p_{lt} \), \( p_{ao} \), \( p_{r} \), LNR )
    Else
        Lisogenic (c, CP, plt, LIT)
    End If
    Fbest = min (F)
Next
Loop Until iterations = Nmax or check_gap(CP ) = True
End VS_SMTWTP

procedure Lytic_Replication (c, CP, \( p_{lt} \), \( p_{ao} \), \( p_{r} \), LNR )
    CS = CP(c) /* Current_Solution
    /* Get the value of nucleus capsid
    z = Get_Probability_of_Random_Binomial (LNR, \( p_{r} \))
    CP(c).NR = CP(c).NR + z
    CS.LNR=LNR(F(c)-Fbest)/Fbest
    /* Check infection
    if CS.NR > CS.LNR then
        /* Arrange descendly solutions of neighbourhood based on health of cells
        Vs = Neighbourhood (CS)
        VAS = Get_Arranged_Neighbourhood(Vs)
        /* Arrange CP ascendingly based on health of cells
        CPA = Get_Clinical_Picture_Arranged (CP)
        i = 1
        for each \( S' \in VAS \)
            if i <= |CPA| and i <= |VAS| then
                a = random_value
                b = random_value
                if a < \( p_{l} \) and b > \( p_{ao} \) then
                    /* Replace CPA(i) with a new solution CS’
                    CPA(i) = CS’
                    CPA(i). Replication_type= Get_Randomly_Replication_Type (\( p_{lt} \))
                    FA(i) = SMTWTP (CPA(i), data)
                    replace = true
                end-if
                i = i + 1
            end-if
        end-for
    end-if
end Lytic_Replication

procedure Lysogenik_Replication(c, CP, plt, LIT)
    CS = CP(c)
    CS.IT = CS.IT + 1
    CS.LIT=LIT(F(c)-Fbest)/Fbest
    if CS.IT > CS.LIT' then
        s = Get_Random_Gen ()
        /* Do Mutatuin CS
        CSNEW = Mutation(CS, s)
        CSNEW.Replication_Type = Get_Replication_Type (\( p_{lt} \))
        F(c) = SMTWTP (CSNEW, data)
    End-if
end Lysogenic_Replication
5. Numerical Experiment

The experiment was conducted using some data sets obtained from OR Library. The data sets consists of 125 instances of 40 jobs, 12 instances of 50 jobs and 25 instances of 100 jobs. The experiment was done by varying the values of parameters using in the algorithm on the case of 40 jobs. Best parameters combination obtained from case of 40 jobs then was applied on other cases.

5.1. Parameters of the algorithm

The experiment was conducted by alternating the values of the parameters. Table 1 shows the parameters values used in the experiment.

| Parameter | Scenario | 1 | 2 | 3 | 4 | 5 | 6 |
|-----------|----------|---|---|---|---|---|---|
| CPS       |          | 10| 20| 30| 40| 50| 60|
| Plt       |          | 0 | 0.2| 0.4| 0.6| 0.8| 1 |
| Pi        |          | 0 | 0.2| 0.4| 0.6| 0.8| 1 |
| Pan       |          | 0 | 0.2| 0.4| 0.6| 0.8| 1 |
| Pr        |          | 0.2| 0.4| 0.6| 0.8| 1 | -|
| LNR       |          | 5 | 10| 15| 20| 25| 30|
| LIT       |          | 5 | 10| 15| 20| 25| 30|
| Mutation  |          | swap| Flip| fslide| bslide| random| combination |

The basic parameters that are used to test the performance are:
CPS = 20, plt = 0.6, pi = 0.7, pan = 0.3, pr = 0.7, LNR = 5, and LIT = 5. To get best parameters we ran the algorithm on the case of 40 jobs. The best parameters obtained in this case then applied for all other problems. Each parameter was tested by running the algorithm 10 times in order to evaluate the performance of the algorithm. Then, we varied the parameters in order to get better results in terms of minimum total weighted tardiness.

5.1.1. Clinical Picture Size (CPS)

CPS indicates the size of the solutions contained in the Clinical Picture (CP). In the initialization phase, CP is obtained randomly based on CPS value. The influence of CPS on the performance is shown in Fig 1.
Based on the results shown in Figure 1, the greater the value of CPS, the better the searching process in achieving optimum solution. In terms of computing time, the higher CPS value requires longer time to do searching process. Thus, we cannot make this CPS as high as possible, otherwise we require very high computing time. The value of CPS of 40 is reasonably good.

5.1.2. Lytic Probability (plt)
Probability of lytic replication (plt) shows the proportion of lytic and lysogenic replication in the sample, which is analogous to the neighborhood and mutation. If the value of plt is greater than 0.5, then the searching process with neighborhood is more dominant than mutation. This triggers obtaining the optimal values, but may be local solutions. If the value of plt is less than 0.5, then the searching process is using more mutation than neighborhood. In this way, the search process is more slowly but can potentially reach the global optimum. It might get out of the local optimum trap.

Based on the experimental results shown in Figure 2, we see that for the data set with 40 jobs, the best plt is about 0.4. Therefore, a good value of the plt make search faster and can reach a global optimum. Based on this experiment, the combination of two types of search (neighborhood and mutation) provides a good result.

5.1.3. Infection probability (Pi) and antigen (Pan)
The probability of infection (Pi) indicates the probability of infection of the virus to the cell that was around cells previously seized. While, the probability of antigen (Pan) indicate the probability of cells secrete antibodies to against virus infection. With the same value of Pan (Pan = 5), the experiment that has been done for Pi values between 0.2 to 1, did not indicate significant influence on objective value. However, if the value of Pi = 0, the results obtained are very bad. The value of Pi = 0 indicates the analogy that no single virus that infects neighboring cells, so there is no neighborhood. In terms of computing time, the greater the value of Pi, the longer the computing time. With the same value of Pi (Pi =0.5), the experiment has been done with Pan values between 0 and 0.8. The change of Pan does not indicate significant influence on objective value. However, if the value of Pan = 1, the results...
obtained are very bad. \( \text{Pan} = 0 \) indicates the value of the analogy that all the neighboring cells can prevent infection, so the neighborhood was not the case. In terms of computing time, the greater the value of \( \Pi \) computation time becomes faster. The value of \( \Pi \) and \( \text{Pan} \) has an effect on the size of the neighborhood around solutions that are previously been broken. The bigger \( \Pi_n \), the more the number of neighborhood.

### 5.1.4. Single probability of one replication (\( pr \))

The Single probability of one replication is based on an analogy to the probability that nucleus capsid inside the cell to replicate. The greater value of \( pr \) make number of nucleus capsid be more developed. This triggers the virus to quickly solve the resulting cell neighborhood. At each iteration, increasing value of \( \text{NR} \) for random values obtained from \( pr \) and value of \( \text{LNR} \). Therefore, this parameter affect how often neighborhood occurs.

### 5.1.5. \( \text{LNR} \)

\( \text{LNR} \) value is the limit value of \( \text{NR} \). If the \( \text{NR} \) value exceeds \( \text{LNR} \), then nucleus capsid within the cell will break the cell wall and out of the cell to carry out an infection to another cells that was around. This is analogized to the new solution approach to the previous solutions. \( \text{LNR} \) value of a cell is also analogous to the health of the cell. In this algorithm, the value of \( \text{LNR} \) cells is obtained from the value of the function relative to the best function value multiplied by the basic \( \text{LNR} \). Parameter \( \text{LNR} \) essentially does not make a significant effect on the algorithm, since the development of the \( \text{NR} \) value is influenced by the amount of \( pr \) which is also based on the value of \( \text{LNR} \).

### 5.1.6. \( \text{LIT} \)

\( \text{LIT} \) value indicates the limit value of \( \text{IT} \). If the \( \text{IT} \) value exceeds \( \text{LIT} \) value, then the nucleus capsid within the cell can be mutated. This is analogized to a mutation of the previous solution to the new solution. \( \text{IT} \) value is incremented by one on each iteration, while the value derived from the value of \( \text{LIT} \)-cell function relative to the value of the best functions of the base multiplied by the \( \text{LIT} \). \( \text{LIT} \) This parameter determining the frequency of mutations.

### 5.1.7. Mutation

In the case of combinatorial, mutation can be done in several ways that include swap, flip, and slide. In this research, the experiment perform six scenarios of mutations, swap, flip, slide forward, slide backward, random mutation, and combination mutation. Based on preliminary experiment on alternating mutation type, swap mutation type provides the best solution. Using a combination of the mutation types also produce a good result but taking a longer time.

Based on parameter setting experiment, the best parameters combination is presented in Table 2.

| No | Data   | \( \text{Nmax} \) | \( \text{CPS} \) | \( \text{Plt} \) | \( \text{pi} \) | \( \text{Pan} \) | \( \text{Pr} \) | \( \text{LNR} \) | \( \text{LIT} \) |
|----|--------|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 2  | 40 jobs| 1000            | 20          | 0.6         | 0.7         | 0.3         | 0.7         | 5           | 5           |
| 4  | 50 jobs| 1000            | 20          | 0.6         | 0.7         | 0.3         | 0.7         | 5           | 5           |
| 5  | 100 jobs| 2000           | 20          | 0.4         | 0.7         | 0.3         | 0.7         | 5           | 5           |

### 5.2. Results

Based on the experiment results, Viral System can solve optimally 125 out of 125 instances with 40 jobs. A total of 123 instances were solved optimally using the maximum iteration 500, while for 2 instances, (instances 40_58 and 40_85) the optimal solutions were found the maximum iteration 1000. Computing time required on average is 23.7 sec. For those problems with 50 jobs, 101 out of 125 instances can be solved optimally. Computing time required on average is 41.6 sec. For problem with 100 jobs, 9 out of 25 instances can be solved optimally. Computing time required on average is 344 sec. We notify that the greater the number of jobs, the harder to find optimal solutions.
used in the algorithm should be tuned correctly in order to have the best performance. The results is compared with some algorithms applying on the same problems [1].

| Algorithm | 40 job | 50 job | 100 job |
|-----------|--------|--------|---------|
| SAmulti-s | 121    | 115    | 59      |
| TS        | 115    | 111    | 96      |
| TSmulti-s | 118    | 113    | 103     |
| CE_ST     | 124    | 118    | 80      |
| ILS-dyna  | 123    | 125    | 82      |
| VS        | 125    | 101    | 9 *     |

*For case with 100 jobs, we only ran experiment with 25 instances

The results show that VS is promising in solving scheduling problem. But for big problem the computing time is still high.

6. Conclusion
Viral Systems has been successfully applied for solving Single Machine Total Weighted Total Tardiness Problem (SMTWTP). The experimental results show that Viral System solve optimally 125 of 125 instances of 40 jobs, 101 of the 125 instances of 50 jobs, and 9 of the 25 instances of 100 jobs. The increasing of number of jobs affect decreasing of the number of instances solved. Viral systems algorithms have 8 parameters that have an influence on the search process. For the problem with large number of job, intensive experiment is needed and modification on the algorithm might improve the performance.

REFERENCES
[1] Borgulya, I. (2002). "A Cluster-based Evolutionary Algorithm for the Single Machine Total Weighted Tardiness-scheduling Problem", Journal of Computing and Information Technology - CIT 10, 2002, 3, 211–217.
[2] M. L. Pinedo, Scheduling Theory, Algorithms, and Systems. New York: Springer, 2008.
[3] H. Nazif, A Genetic Algorithm on Single Machine Scheduling Problem to Minimise Total Weighted Completion Time, European Journal of Scientific Research, vol. 35, pp. 444-452, 2009.
[4] J. K. Lenstra, A. H. G. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," Annals of Discrete Mathematics, vol. 1 pp. 343–362, 1997.
[5] C. N. Potts and L. N. V. Wassenhove., A branch and bound algorithm for the total weighted tardiness problem, Operations Research, vol. 33 pp. 363–377, 1985.
[6] R. K. Congram, C. N. Potts, and S. L. v. d. Velde, An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem, INFORMS J. on Computing, vol. 14, pp. 52-67, 2002.
[7] X. Wang and L. Tang, A Population-Based Variable Neighborhood Search for The Single Machine Total Weighted Tardiness Problem, Computers and Operations Research, vol. 36, pp. 2105-2110, 2009.
[8] P. Cortes, J. M. Garcia, J. Munuzuri, and L. Onieva, Viral systems : A new bio-inspired optimisation approach, Computers & Operations Research, vol. 35, pp. 2840 - 2860, 2008.
[9] P. Cortes, J. M. Garcia, J. Munuzuri, and J. Guadix, A viral system massive infection algorithm to solve the Steiner tree problem in graphs with medium terminal density, Industrial Engineering, vol. 2, pp. 71-77, 2010.
[10] Santos, B and P. Willy, Metode Metaheuristik, Konsep dan Implementasi. Surabaya: Guna Widya, 2011.
[11] Suryadi, D and E. K. Kartika, Viral Systems Application for Knapsack Problem, Proceedings of the 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, 2011.
[12] Santos, B. and M. A. Budiman, Simple Algorithm for Single Machine Total Weighted Tardiness Problem," in Proceeding of Industrial Engineering and Service Science, 2011