Comp. Meth. Civil Eng., Vol.4 No.1 (2013) 65 - 82

CMCE
Computational Methods in Civil Engineering

Enhanced ACS algorithms for plastic analysis of planar frames

M. Jahanshahi [a,*], M. Pouraghajan[a], M. Pouraghajan[b]
[a]*Department of Civil Engineering, School of Science and Engineering, Sharif University of Technology, International Campus, Kish Island, P.O. Box 76417-76655, Iran*
[b]*Department of Computer Engineering, Mazandaran University of Science and Technology, Babol, Iran*

Received 21 October 2013; accepted in revised form 14 April 2014

Abstract

In recent years, the trend in solving optimization problems has been directed toward using heuristic algorithms such as neural networks, genetic and ant colony algorithms. The main reason for this trend can be attributed to the fact that these algorithms can be efficiently adjusted to the specific search space to which they are applied and consequently they can be used for many optimization problems of different nature. In this paper, the behavior of classical ACS algorithm in finding collapse load factor of two-dimensional frames is investigated closely. Time consuming and redundant parts that greatly affect the performance are removed leading to an accelerated ACS algorithm called variant one in this work. For some frames with certain combination of plastic moments and loadings the first variant does not lead to acceptable results. Therefore a few constraints intended to accelerate the variant one of ACS algorithm are eliminated and some provisions are added to bias the solution toward ant decision making strategy rather than problem dependent information. Consequently a new variant called variant two is proposed that can be used for a wider range of frames with of course more computational effort.

Keywords: collapse load factor; collapse mechanism; plastic limit analysis; ant colony optimization; heuristic methods

1. Introduction

The minimum and maximum principles are the basis of nearly all the analytical methods used for plastic analysis and design of frames [1]. The most frequently used method, based on the minimum principle, is the combination of elementary mechanisms, first developed by Neal and Symonds [2–4].

The problem of plastic analysis and design of frames with rigid joints was solved by linear programming by Charnes and Greenberg [5], as early as 1951. Further progress in this field is attributed to Heyman [6], Horne [7], Baker and Heyman [8], Jennings [9], Watwood [10], Gorman [11], Thierauf [12], Kaveh [13], Kaveh and Khanlari[14], Munro [15] and Livesley [16] among others. The progress during 1955–1977 is well documented in the papers of

[*] Corresponding author.
Email Address: jahanshahi@sharif.edu

reference [17], and a survey of research results achieved in the subsequent 25 years on limit analysis and design in plasticity is provided by Maier et al. [18].

Plastic analysis and design of frames using combination of elementary mechanisms encompass limitations, which prevent it from being used as an efficient tool for custom analyses. As an example, the number of mechanisms that have to be considered and the tedious work of combining them to find the collapse mechanism can be mentioned as one of these limitations. Despite the problems mentioned above, it is important to develop an algorithm that computes the collapse load factor and the corresponding mechanism with adequate accuracy within the expected time span. Thus the necessity and role of a heuristic algorithm such as ACS becomes more and more important [26]. In recent years, the trend in solving optimization problems has been directed toward using heuristic algorithms such as neural networks, genetic and ant colony algorithms. The main reason for this trend can be attributed to the fact that these algorithms can be efficiently adapted to the specific search space to which they are applied and consequently they can be used for many optimization problems of different nature. In this paper, the behavior of classical ACS algorithm in finding collapse load factor of two-dimensional frames (see reference [26]) is investigated closely. Time consuming and redundant parts that greatly affect the performance are removed leading to an accelerated ACS algorithm called variant one in this work. For some frames with certain combination of plastic moments and loadings the first variant does not lead to acceptable results. Therefore a few constraints intended to accelerate the variant one of ACS algorithm are eliminated and some provisions are added to bias the solution toward ant decision making strategy rather than problem dependent information. Consequently a new variant called variant two is proposed that can be used for a wider range of frames with of course more computational effort.

2. GENERATION OF ELEMENTARY MECHANISMS

In order to find a set of independent mechanisms, the method developed by Watwood [10] can be used. However, in this method joint mechanisms are computed as well, which is unnecessary from computational point of view because joint mechanisms automatically can be assigned to each joint.

Moreover, axial deformations can also be neglected, since mechanisms are the result of excessive deformations in rotational degrees of freedom leading to plastic hinges. With these assumptions, one ends up with a method similar to that of Pellegrino and Calladine [20] and Deeks [21].

Figure 1 show a typical member, in which the elongation of each member is specified in terms of its end displacements, leading to the following equation:

$$e = \left(d_{xk} - d_{xi}\right) cos\,\alpha + \left(d_{yk} - d_{yi}\right) sin\,\alpha \tag{1}$$

Writing the same expression for all members, leads to the following equation:

$$\boldsymbol{e} = \boldsymbol{C}\boldsymbol{d} \tag{2}$$

In a valid mechanism, members do not elongate, therefore in order to find basic mechanisms it is necessary to solve an equation for which the elongation vector is zero, i.e. we should have:

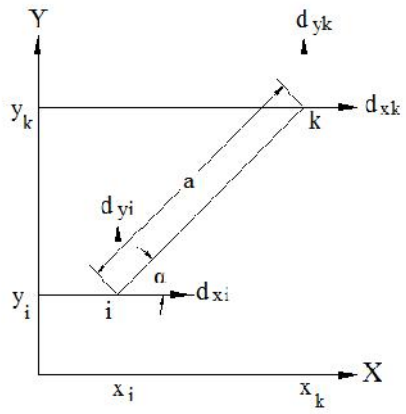

Figure 1. End displacements of a typical member

$$\boldsymbol{Cd} = \boldsymbol{0} \tag{3}$$

Since the assembly is not a truss, it is not stable with all joints pinned. Therefore, the difference between the number of columns of the coefficient matrix $\boldsymbol{C}$ (i.e. the compatibility matrix relating the elongations of members $\boldsymbol{e}$ to the displacements of joints $\boldsymbol{d}$) and the number of rows is equal to the number of independent mechanisms (i.e. the dimension of null space of $\boldsymbol{C}$ [22]). By performing Gaussian elimination, Eq. (3) transforms to:

$$\left[\boldsymbol{I} \vdots \boldsymbol{C}_d\right]\begin{Bmatrix}\boldsymbol{d}^i \\ \boldsymbol{d}^d\end{Bmatrix} = \boldsymbol{0} \tag{4}$$

Thus, columns corresponding to independent displacements $\boldsymbol{d}^{\mathrm{i}}$ are reduced to the identity matrix $\boldsymbol{I}$, the order of which shows the dimension of row space or column space of $\boldsymbol{C}$. By rearranging the terms in Eq. (4), $\boldsymbol{d}^{\mathrm{i}}$ can be expressed in terms of $\boldsymbol{d}^{\mathrm{d}}$ as

$$\boldsymbol{d}^i = -\boldsymbol{C}_d\boldsymbol{d}^d \tag{5}$$

Choosing as many dependent vectors for $\boldsymbol{d}^{\mathrm{d}}$ as the number of independent mechanisms and computing the independent vector $\boldsymbol{d}^{\mathrm{i}}$ using Eq. (5) leads to a solution to Eq. (3). To simplify the computational approach, dependent vectors can be constructed each time by setting one of the dependent displacements to unity and the other displacements to zero. Details of such an approach are available in reference [20].

Following Deeks [21], independent mechanisms can be purified by removing the excess hinges to obtain a set of potential collapse mechanisms. Using this method, for each independent mechanism in turn it is checked whether that mechanism contains a complete set of active hinges of another independent mechanism. If this is the case, it is purified by removing the contained mechanism. This process is repeated until no modification is possible.

3. DETERMINATION OF COLLAPSE LOAD FACTOR

The collapse load factor is calculated using the virtual work theorem. Rotations and displacements are considered to be virtual and the internal and external works are computed using this assumption. The collapse load factor is then the ratio of internal virtual work to external virtual work for a specific mechanism, i.e.

$$\lambda_c = \frac{Int.W}{Ext.W} \tag{6}$$

The external virtual work is the sum of the products of all joint forces $\boldsymbol{P}$, and the corresponding joint displacements, $\boldsymbol{d}$, in the direction of these forces:

$$Ext.W = \boldsymbol{P}^T \boldsymbol{d} \tag{7}$$

The internal virtual work is computed by adding up all the rotations at active hinges multiplied by the plastic moments of members in which active hinges are formed. However, since the plastic moments resist the rotations at hinges, the internal work is always positive and therefore the absolute values of rotations can be used.

$$Int.W = \boldsymbol{M}_p^T \left|\boldsymbol{r}\right| \tag{8}$$

Since joint mechanisms have been neglected during the formation of independent mechanisms, it is necessary to find the location of hinges in members. These are determined by minimizing the internal virtual work. If a joint is restrained against rotation, hinges are formed in all members connecting to that joint. However, if the joint is not restrained against rotation, hinges are formed in (n – 1) members among the n members connected to that joint. In this case, n possible locations exist for hinges and it is necessary to find a location which minimizes the internal virtual work. When less than the maximum number of hinges form, the rotation in one or more of the assumed hinges is zero and does not contribute to the virtual work [21].

4. ANT COLONY SYSTEM

Ant colony optimization (ACO) is a recently developed optimization algorithm and ant colony system (ACS) is a variant of the former which has been shown to behave more robustly and provide better results. In this work ACS is chosen as the underlying optimization algorithm for finding the collapse load factor of frames. In the sequel of adapting ACS to the problem of finding the minimum collapse load factor, it is inevitable to give a brief description of ACO, and provide justifications for implementing ACS [26].

The building blocks of ACO and ACS are cooperative agents called *ants* [22]. These agents encompass simple capabilities, which make their behavior look like real ants. Ants mark paths which lead them to food sources by depositing pheromone and they communicate information through pheromone trails that they leave behind. However it should be noted that pheromone evaporates and its effect weakens over time. As a result of pheromone accumulation and evaporation, more ants tend to pass over certain paths and these paths are visited more often as the intensity of pheromone increases.

As an illustrative example, consider the sketch shown in Figure 2. Assume that there are two paths along which ants can move from the nest to the food source, and vice versa. Also assume that there are 30 ants. At first there is equal probability for ants to select paths, therefore 15 ants select one path and the other 15 select the other path. Since the ants selecting the shorter path reach the food source sooner than others, the result is that any ant returning back to the nest finds more pheromone laid on the shorter path both by half of the ants that initially selected this path and those that have already returned to the nest. Consequently, the number of ants selecting the shorter path increases as the amount of pheromone being laid on this path increases over time.

In order to get more insight into the problem, suppose that the length of the longer path is two times the length of shorter one. It is desired to know what happens at discrete time steps t = 1, 2 . . . Assume that at t = 1, 30 ants start to move to the food source either through the shorter path or through the longer one. Also assume that each ant moves at a velocity of 1 per time unit and lays down a pheromone trail of intensity 1 that evaporates completely after each time step.

At t = 1 there is no trail on paths. The probability of choosing either paths is equal. At t = 2 the new 30 ants that are headed for the food source find a trail intensity of 15 on the longer path laid by the 15 ants that chose this path and an intensity of 30 on the shorter path laid by the other 15 ants and the 15 ants returning back to the nest through this path. The probability of choosing the shorter path is therefore doubled according to the amount of pheromone being laid. This process is continued until all of the ants eventually select the shorter path.

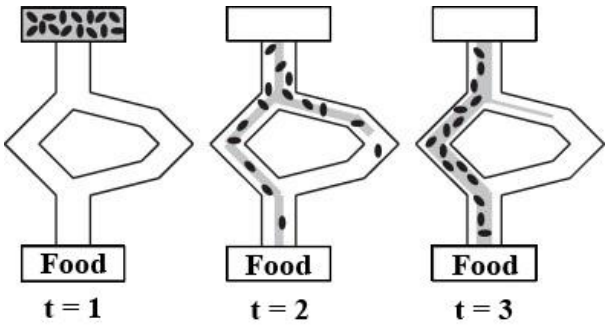

Figure 2. The mechanism of pheromone laying in paths leading to food source

This brief discussion gives an idea about the behavior of real ants and the mechanism based on which optimization algorithms such as ACO are developed. A thorough description of ACO and its descendant, the ACS algorithm can be found in reference [23, 24] and its applications to plastic analysis of frames in reference [26].

4.1. Adopting ACS for finding minimum collapse load factor

Assuming that a typical frame has n elementary mechanisms, a graph consisting of n nodes is constructed and each node is associated to each elementary mechanism. The set of nodes are connected together using $n(n-1)/2$ edges resulting in a clique (see references [13,19,25] for definitions). A predefined number m of ants are randomly distributed on the nodes of graph and the search starts by moving the set of ants from their current position to newly selected nodes based on a decision making rule. According to local search strategies that have been adopted each ant normally visits n nodes during its tour and consequently nm movements take place in each iteration.

At the end of each iteration, the best ant in the iteration or the best-so-far ant updates the pheromone matrix. After a total of N iterations have been exhausted, the best-so-far ant is the representative of the suboptimal solution obtained by the algorithm. There is no necessity for the number of ants to be a whole multiple of the number of independent mechanisms. Although increasing the number of ants can increase the diversity of solution, but as the size of the problem is increased, the computational effort required to solve the problem increases more rapidly. Limiting the number of ants to a predefined number and distributing them randomly over the nodes has proven to be efficient. Ants have memory in the sense that they remember the history of mechanisms being combined. In other words, they save the

mechanism resulting from the combination of the one that they already have saved with the new independent mechanism corresponding to the node that they are bound to. Ants move based on pseudorandom proportional rule from one node to another (see reference [24] for general definition of pseudorandom proportional rule and references [26,27] for its application to plastic analysis of frames). Consider two nodes *i* and *j* and the combination of mechanism stored by the ant located on node *i* with the independent mechanism corresponding to node *j*. Denoting the load factor of ant's mechanism by λ_i, of independent mechanism associated to node *j* by λ_j and of combined mechanism by λ_c, the distance from node *i* to node *j* can be computed as follows, [26]:

$$d_{ij} = \lambda_{max} + \lambda_c - \lambda_i \tag{9}$$

where λ_{max} is determined so that the distance from node *i* to all admissible neighbors is positive. This definition of distance is modified in the following sections to improve the performance.

4.2. Ant Colony Algorithm

In reference [27], it was shown that genetic algorithm finds the minimum collapse load factor faster than ant colony algorithm but this load factor is less accurate comparing to the one resulting from the application of ant colony algorithm. The idea behind optimizing the ant colony algorithm applied to the problem of plastic analysis of frames is to modify the algorithm in such a way that unnecessary steps affecting the performance of algorithm are removed. This way, without deteriorating the accuracy of algorithm we can obtain the same result in less time.

The pseudo code related to the previous implementation of ant colony algorithm is shown in the following figure. After initializing the fundamental variables and objects, the main loop begins with positioning the ants on the nodes of construction graph. In this loop, ant colony operations are performed as many times as required. Two nested loops operate inside the main loop. The first loop (which iterates equal to the number of available mechanisms) is intended to associate each ant to each mechanism. The second loop (which iterates equal to the number of mechanisms minus one) is intended to make each ant visit the remaining nodes (or mechanisms) except the one on which the ant is initially positioned.

In each and every single move from one node to another, the following steps are accomplished.

- The current mechanism which is associated to the ant is combined by all the remaining mechanisms one by one.
- For each combination the difference between the current position of the ant and the next potential position of it is calculated.
- For each combination, the probability of moving from the current node to the next potential node is calculated.

After all the combinations and calculations performed in a single move, the algorithm generates a random number between 0 and 1 and compares this value with the contents of an array called the array of probability values using a simple loop. Each cell in this array corresponds to one node in the graph and stores the probability value of the ant being moved from its current position to this node. If in this comparison, the probability value of a specific node is greater than the generated random number, the algorithm will choose this node as the next node to move to. The current mechanism of ant is combined with the mechanism associate with the selected node and the new load factor is computed accordingly.

```
Procedure Find Collapse Mechanism
        Initialize Data
        While (termination condition not met) do
            Position Ants on Individual Mechanisms
            Construct Solution
            Update Pheromone Matrix
            Save the Best Combined Mechanism
            Remove Ants
         End while
End procedure
Procedure Initialize Data
        Initialize Ants
        Initialize Pheromone Matrix
End procedure
Procedure Construct Solution
        For i = 1 to number of iterations
            For j = 1 to number of mechanisms
                For k = 1 to (number of mechanisms - 1)
                    Move Ant Based on Decision Rule (j)
                End for
             End for
         End for
End procedure
```

Figure 3. Pseudo code related to previous implementation of ant colony algorithm [27]

5. OPTIMIZED ANT COLONY ALGORITHM

The approach described in previous section has advantages and drawbacks. The stability and reliability of results is one of its aspects but there are certain issues with this way of implementing the algorithm which are going to be addressed in this work. The most important point which has to be taken into account is the performance of algorithm and its efficiency in using the computer resources.

The previous implementation of algorithm have the drawback that it does not handle the movements of ants efficiently (see references [26,27]). It is possible that at certain stages of algorithm a few ants are entangled in dead ends making useless the subsequent computational effort devoted to move them to other nodes. This waste of computational resources can greatly affect the performance. The following solution which has been applied to the first variant of algorithm addresses this drawback and improves the performance leading to an algorithm much faster than the previous one. The pseudo code shown in the following figure presents the main difference between the previous implementation and the first variant of algorithm implemented in this work.

```
Procedure Construct Solution
        For i = 1 to number of iterations
            For j = 1 to number of mechanisms
                For k = 1 to (number of mechanisms - 1)
                    If (Previous ant movement was possible) Then
                        Move Ant Based on Decision Rule (j)
```

Else (The ant seems to be stuck to current node)
Break
End for
End for
End for
End procedure

Figure 4. Pseudo code related to the first variant of ant colony algorithm

This code represents a basic change in the hypothesis of algorithm. This means that in the new implementation, unlike the previous one, the movement of ant will be conditional and it depends on the previous moves made by the same ant. Previous implementations do not consider dead ends. A dead end in ant colony terminology stands for a situation in which a given ant is not able or in other words is not willing to move any further and all the remaining nodes seem to be infeasible for the ant to move to. This situation arises when the probability values associated to other nodes are very small implying that the probability of reducing the load factor by combining the current mechanism (contained by the ant) by other elementary mechanisms corresponding to remaining nodes is very low. The ant cannot move to any other node unless the random number generator generates a value close to zero in which case the resulting load factor is hardly lower than the one already contained by the ant.

Previous implementations of algorithm ignore this situation and if they encounter such a case in which the ant cannot move any further in one particular step, they will repeat the whole process for it again and again till the main loop is terminated. The situation can be described more easily using a simple example.

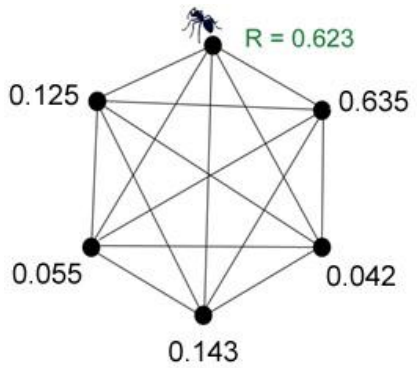

Figure 5. Initial configuration of ACS algorithm for a simple graph

Assume that we are trying to move a given ant on the nodes of the sample graph shown in Figure 5. The ant is initially placed on the top node of the graph and each of the remaining nodes have a number associated to them which stands for the probability of moving the ant from its current position to that node (which has already been calculated by combining two mechanisms and computing the difference value). The value *R* shown in the figure represents the random number generated by the algorithm to be compared against probability values.

As the graph has 6 nodes (or 6 mechanisms associated to them) the innermost loop of the algorithm which is responsible to move the ant through the graph nodes, iterates 5 times (6 - 1) trying to move the ant to all the 5 remaining nodes. In the first iteration after computing the probability values of nodes, a random number is generated and is compared with the probability value for each node. As a result only the closest node at the right side of the ant has a probability value greater than the generated random number. Hence, the ant will move

to this node and it will combine its current mechanism with the mechanism associated to this node and the first iteration is accomplished.

The second iteration begins with the inputs depicted in the following figure. After computing the probability values of the four remaining mechanisms it turns out that the probability of moving the ant in this way is decreasing because the values corresponding to each node are generally degrading to small numbers (due to the low probability of reducing load factor). The generated random number is 0.569 and it is greater than the values assigned to remaining nodes. Consequently no node will be selected for the ant's next move.

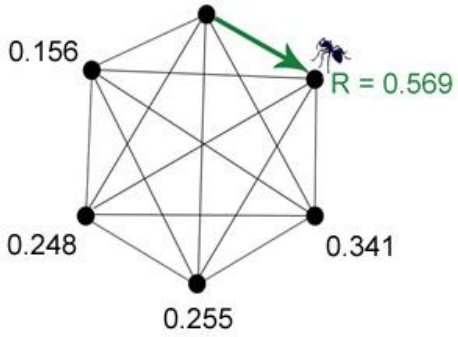

Figure 6. Configuration of ACS algorithm at the beginning of second iteration

What is performed in previous implementations in such a situation is to repeat all the operations already performed in the second iteration for three more times, while the ant is stuck to the same node during the whole operation of algorithm. The implementation of the first variant in this work faces the situation differently. The new approach adds a condition after each step of ant movement. If at any level of iteration no node would be chosen for ant's next movement, the algorithm breaks the current loop and releases the ant starting the process for the next ant movement. This way a huge amount of computational resources and time would be saved helping the algorithm to operate faster. It should however be noted that the elimination of ants occurs only when the pheromone matrix has meaningful entries. In other words, at the first stages of algorithm, ants decide to select different nodes with the same probability. At this time it is very unlikely for a certain ant not to be able to move to another node. After many iterations have been exhausted, pheromone is accumulated on certain paths and evaporates on others. Only then degrading probability values indicate the insufficiency or inefficiency of partial solution made by the ant.

Obtaining a better performance and managing the risk of not losing the accuracy of results also depends on one another change in the basic logic of algorithm. In previous implementations of algorithm, the difference between the load factor of each ant has been used to select the best ant at the end of the iteration (see previous section and also reference [26]); however in this work the final load factor of each ant is used to determine the most efficient ant among them. The idea behind this methodology is supported by the following observations.

- Sometimes, in a particular iteration, an ant cannot move to any other node in the graph for the reasons explained above. This causes the final difference value for this ant to be zero. Therefore it might be identified as the best ant while its final load factor remains constant and possibly higher than other ants.
- As the aim of algorithm is to search for the minimum load factor and the best ant has to maintain that, it is more rational to use the load factor to determine the best ant not even at the end of the algorithm but in each iteration as well.

  Based on these modifications, it is quite expected that a few ants with poor behavior are eliminated from subsequent iterations. Therefore the number of ants distributed over

the nodes should be selected cautiously. If this is the case, the accuracy of final result should not be affected as redundant and time consuming parts of the algorithm are removed only.

In some cases the combination of two elementary mechanisms does not instantly lead to a reduction in resulting load factor. Therefore the strategy selected for the first variant of our algorithm does not work for cases where several elementary mechanisms have to be combined so that a considerable reduction is obtained in the final load factor (see reference [28]). In order to force a given ant to combine elementary mechanisms even when the resulting load factor is not lower than the mechanism already contained by the ant we apply the following modifications for the second variant of algorithm implemented in this work.

- Similar to the first variant, the lowest load factor determines the best-so-far ant. Therefore instead of selecting the best ant based on minimum tour length, the lowest computed load factor is used.
- The meaning of distance is slightly different from the first variant. In the second variant the reduction in load factor of either the mechanism contained by the ant or the one corresponding to the node to which the ant is to move to is the new measure for distance. Denoting the load factor of ant's mechanism by λ_i, of independent mechanism associated to node j by λ_j and of combined mechanism by λ_c, the distance from node i to node j can be mathematically expressed as:

$$d_{ij} = c + \begin{cases} r_{min}, & r_{min} < 0 \\ r_{max}, & r_{min} \geq 0 \end{cases} \tag{10}$$

where r_{min} and r_{max} are minimum and maximum reductions in load factor and are defined as:

$$r_{min} = \lambda_c - max(\lambda_i, \lambda_j) \tag{11}$$

and

$$r_{max} = \lambda_c - min(\lambda_i, \lambda_j) \tag{12}$$

The constant c is chosen such that d_{ij} is always greater than the minimum value used to initialize the pheromone matrix. Using this definition it is obvious that those paths with higher reductions in load factors are more appealing to ants than other paths. Ants have the chance to combine elementary mechanisms although the load factor increases in the first stages of combination.

- The constrain that the ant is not allowed to combine mechanisms when the resulting load factor is increased is removed in this variant thereby granting the chance to the ant to visit all potential nodes and combine the contained mechanism with those corresponding to visited nodes.
- Unlike previous versions, it is not necessary to increase the search space by saving additional combined mechanisms. Instead the same number of nodes as the number of independent mechanisms is used for construction graph.

Strategies similar to the first and third items above have been studied before in the work of Kaveh et al [28]. Merely allowing the mechanisms to be combined does not improve the performance of ant colony algorithm. This is where the role of second modification becomes more important. It ties the decision making strategy of ants to the way they should combine the mechanisms so that a suboptimal collapse load factor is obtained. With these changes, it

is not tried to bias the algorithm toward a specific solution. Therefore it is purely the duty of ACS algorithm to search the space for the suboptimal solution. It is quite expected that the CPU time consumed by this variant is higher than the first one.

6. NUMERICAL EXAMPLES

In this section numerical examples are used to assess the performance of optimized ant colony algorithms. Both the classical and optimized ant colony algorithms are used to solve the examples and results are compared. For each example, the geometry and loading are shown and plastic moments are denoted on individual members. Examples 1 and 2 are solved by the first variant of ACS algorithm and examples 3 and 4 by the second variant. For all examples 20 iterations have been used and ACS parameters have been set to $\alpha = 1.0$, $\beta = 5.0$ and $\rho = 0.5$.

6.1. Example 1

The first example is a four story frame as illustrated in Figure 7. The actual collapse load factor as computed by both algorithms is 2.15. Figure 9 clearly shows that the computation time in each iteration for the first variant of algorithm is smaller than the previous one.

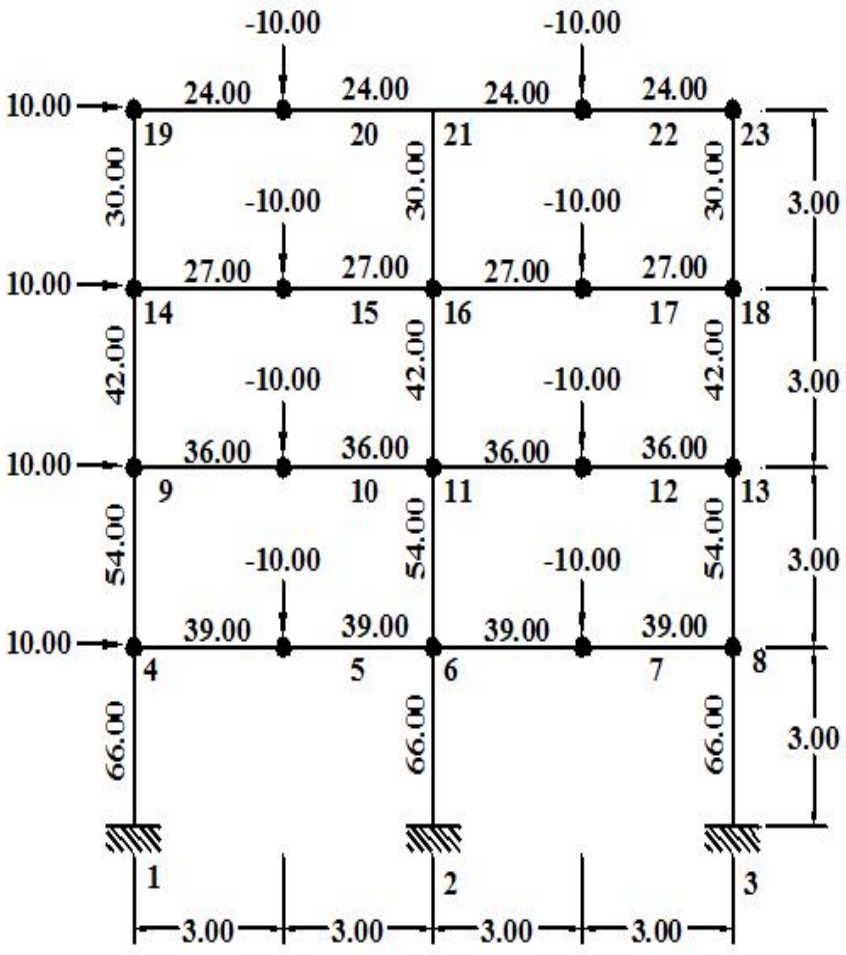

Figure 7. Two-bay, four-story frame for example 1, loading and plastic moments

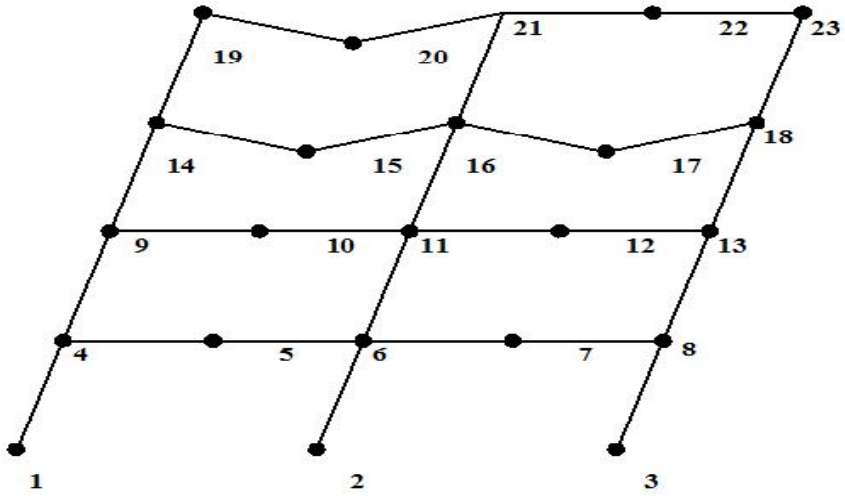

Figure 8. Actual collapse mechanism for the frame of example 1

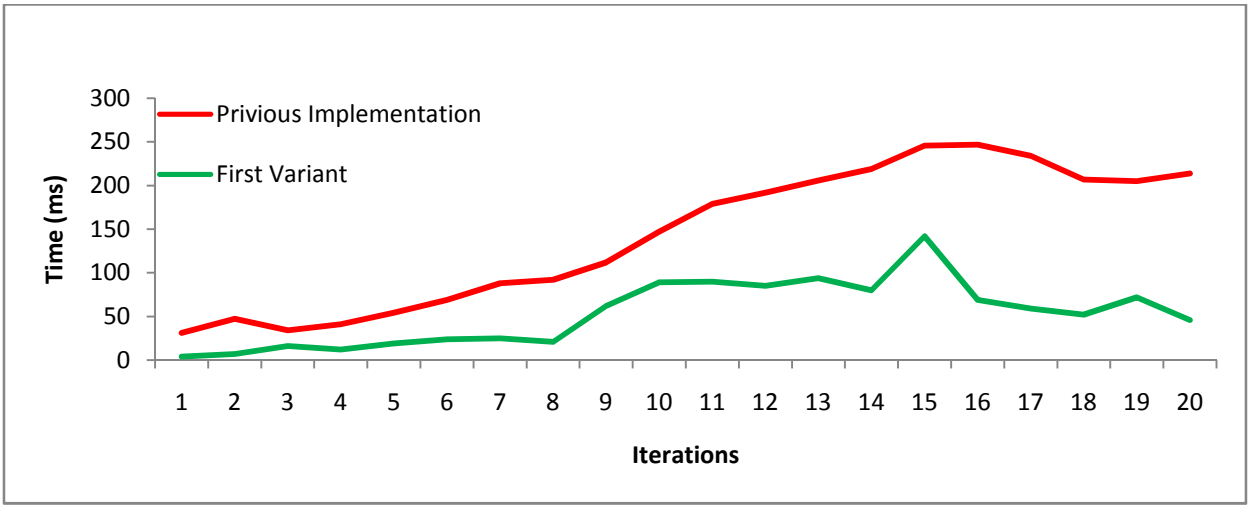

Figure 9. Elapsed for both algorithms applied to the frame of example 1

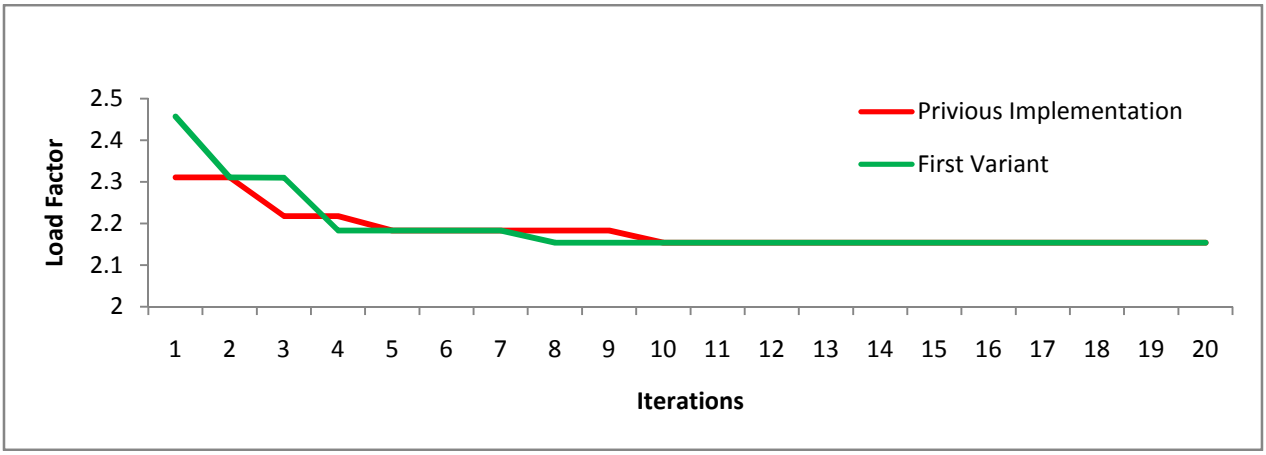

Figure 10. Comparison between load factors computed using both algorithms for the frame of example 1

6.2. Example 2

The second example is a four story frame as illustrated in Figure 11. The actual collapse load factor as computed by both algorithms is 0.65 and both algorithms are converged in second iteration. As in previous example, the computation time in each iteration for the first variant is smaller than the previous implementation and the difference is greater in subsequent iterations.

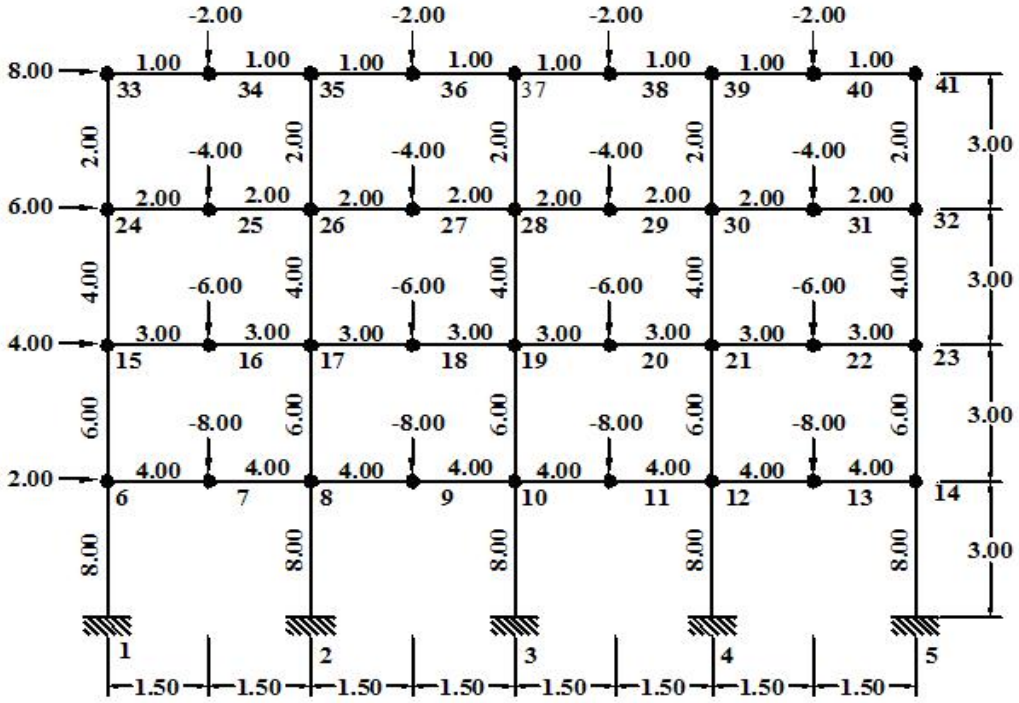

Figure 11. Four-bay, four-story frame for example 2, loading and plastic moments

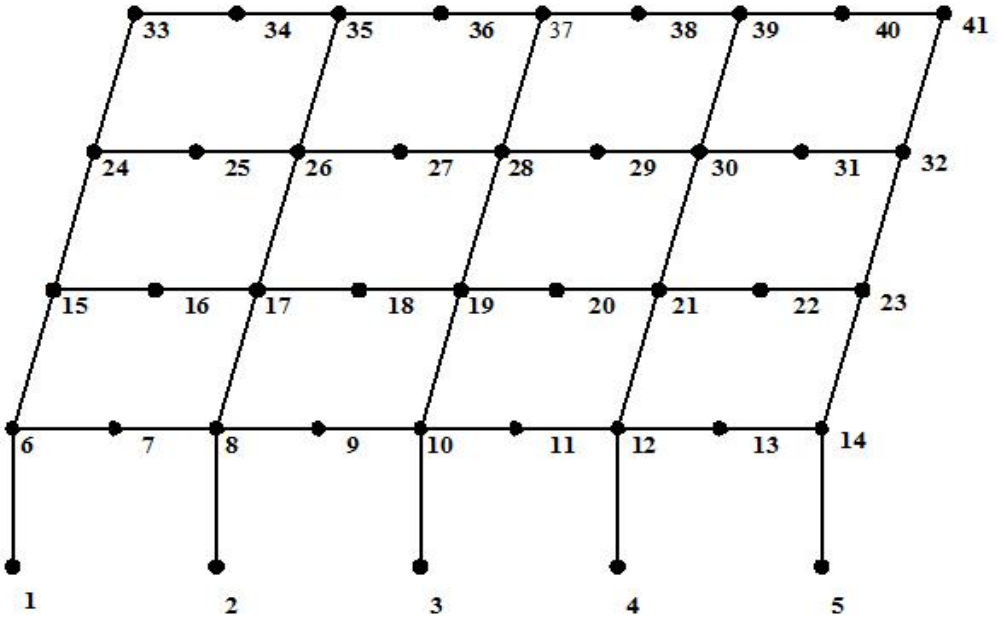

Figure 12. Actual collapse mechanism for the frame of example 2

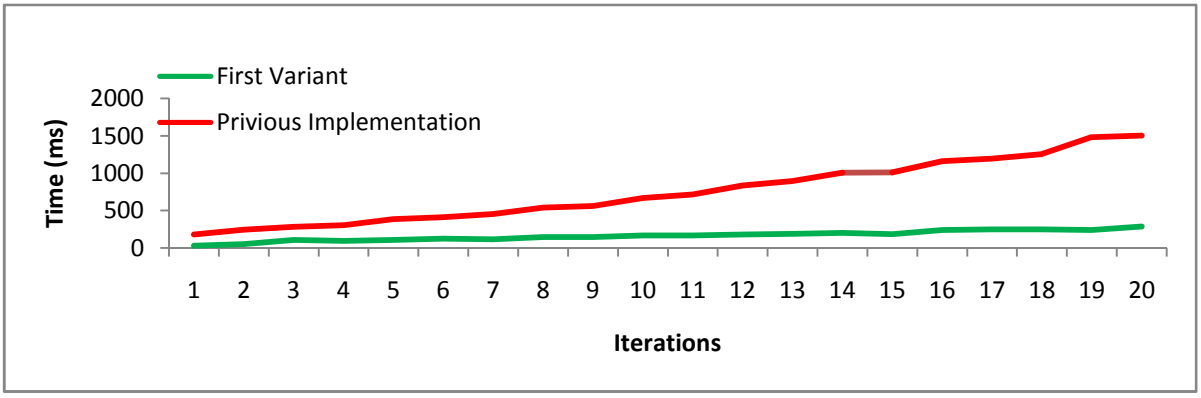

Figure 13. Elapsed time for both algorithms applied to the frame of example 2

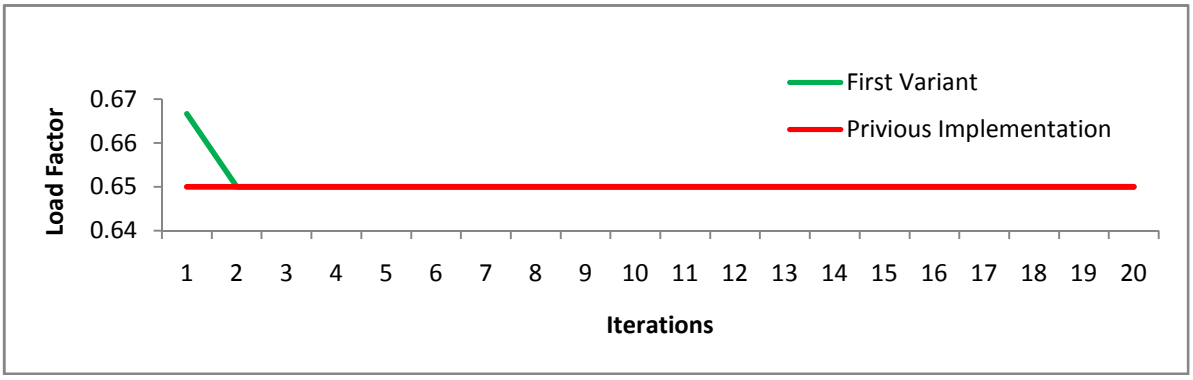

Figure 14. Comparison between load factors computed using both algorithms for the frame of example 2

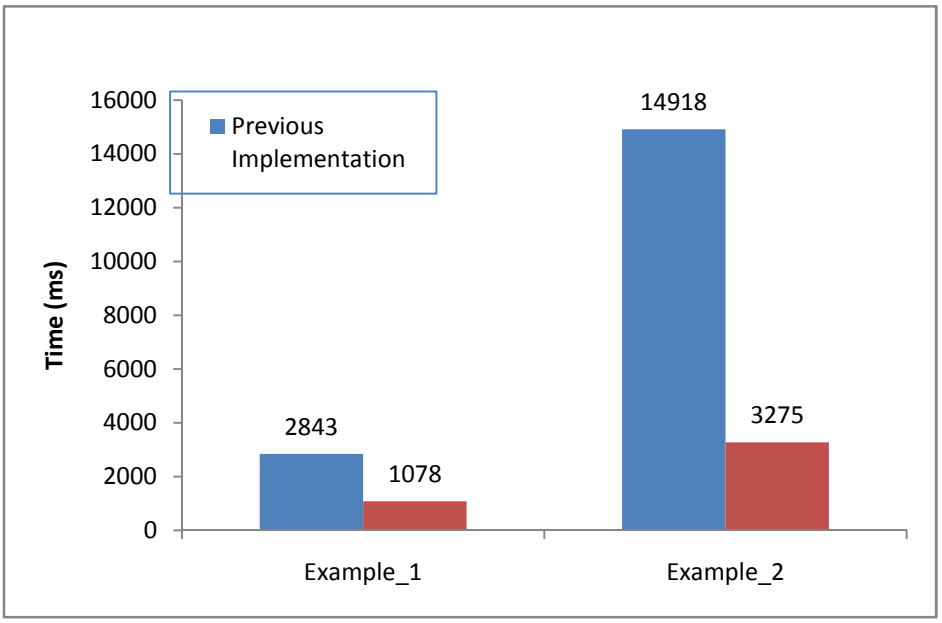

Figure 15. Total elapsed time using both algorithms for frames of examples 1 and 2

The graph in Figure 15 shows the total time elapsed by the previous implementation and the first variant of algorithm for the models of examples 1 and 2. This graph as well as the graphs shown in figures 9 and 13 clearly show that a great deal of computation time

previously devoted to malfunctioning ants is now saved by the algorithm. This saving becomes more pronounced when the number of iterations is increased or the dimensions of problem at hand become larger.

6.3. Example 3

The third example is a frame with three bays and three stories as illustrated in Figure 16. This example has been considered as a special case in reference [28]. The load factor of the actual collapse mechanism is 1.8730. The result obtained from the second variant of algorithm is 1.9242 (about 3% higher).

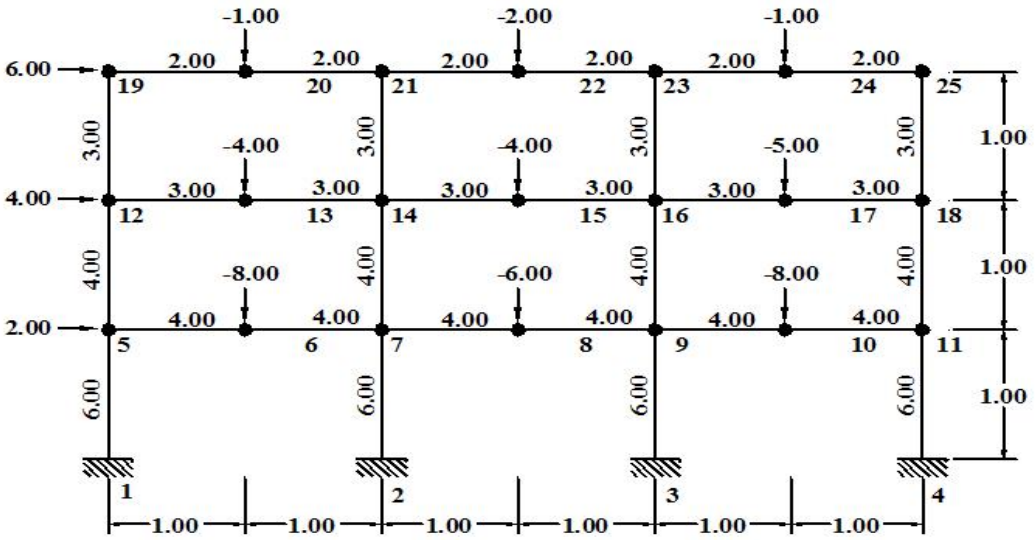

Figure 16. Three-bay, three-story frame for example 3, loading and plastic moments

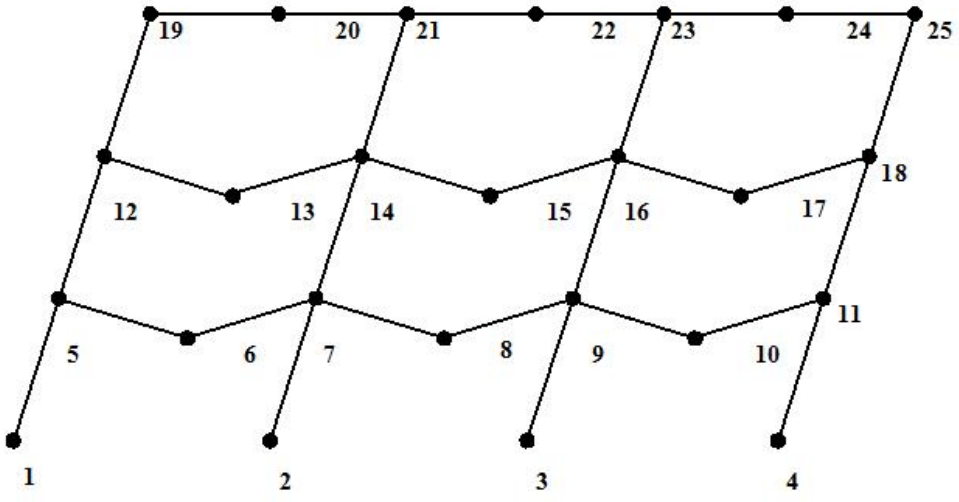

Figure 17. Actual collapse mechanism for the frame of example 3

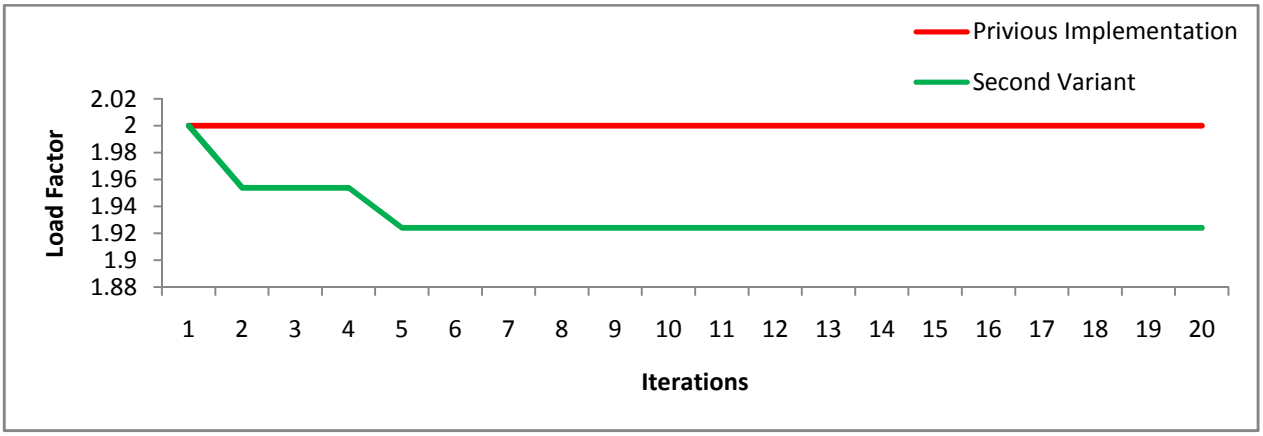

Figure 18. Comparison between load factors computed using both algorithms for the frame of example 3

6.4. Example 4

The forth example is a frame with four bays and four stories as illustrated in Figure 19. This example has also been considered in reference [28]. The load factor of the actual collapse mechanism is 1.7388. The result obtained from the second variant of ACS algorithm is 1.7862.

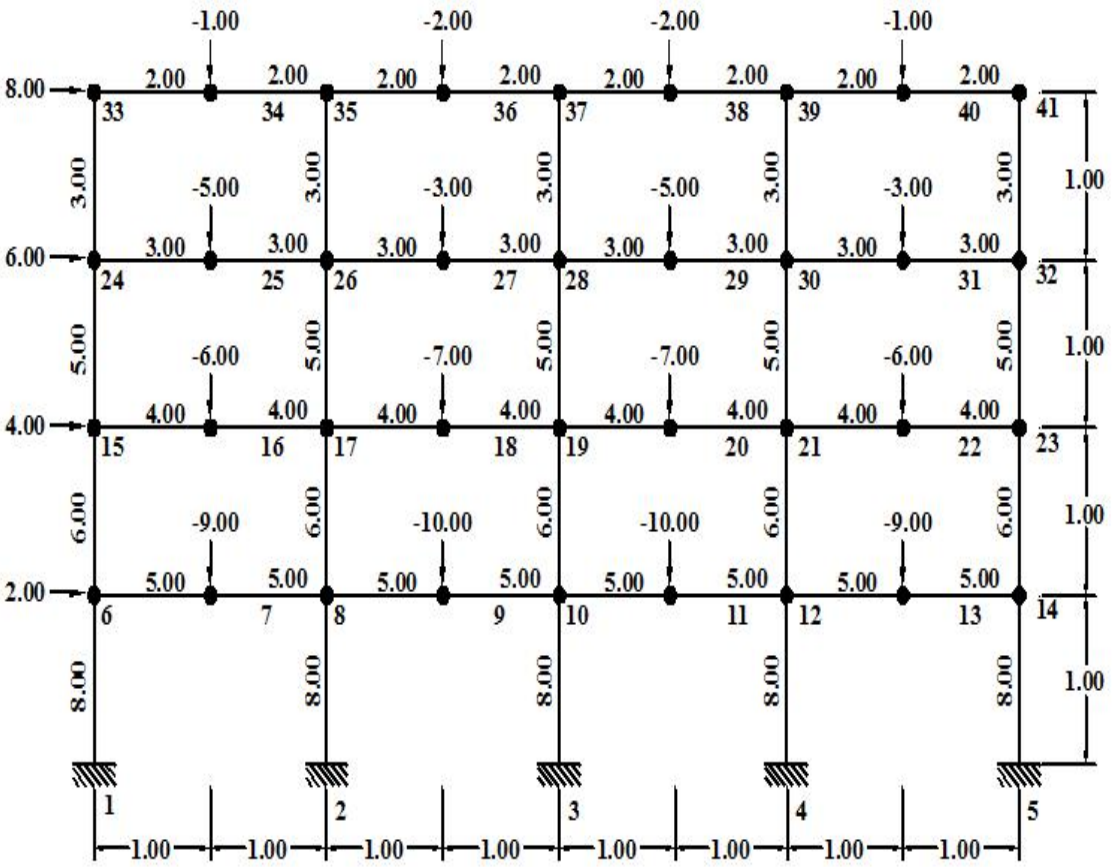

Figure 19. Four-bay, four-story frame for example 4, loading and plastic moments

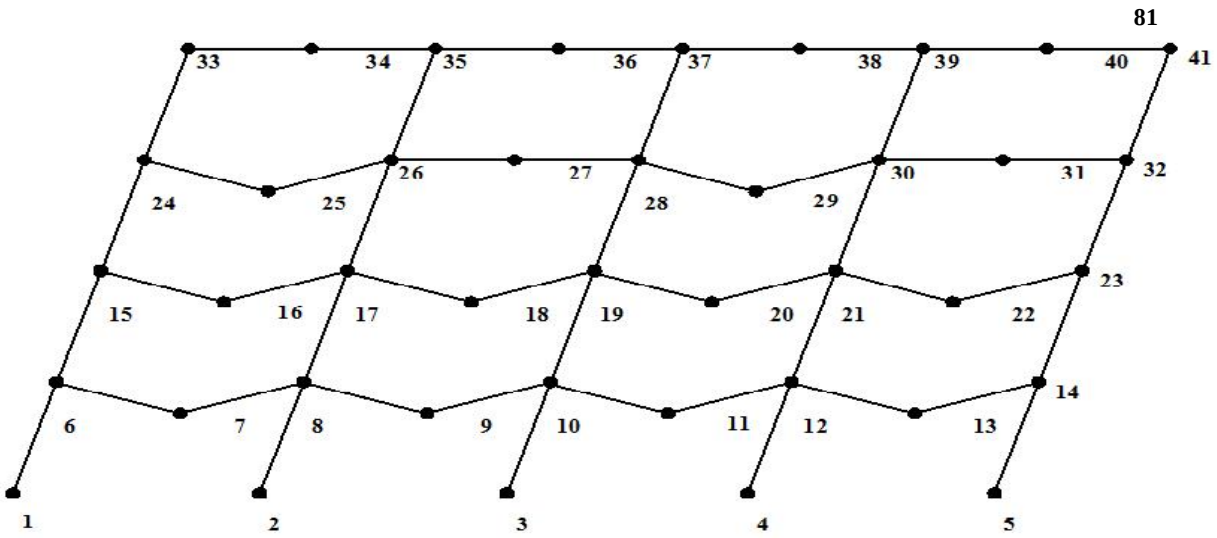

Figure 20. Actual collapse mechanism for the frame of example 2

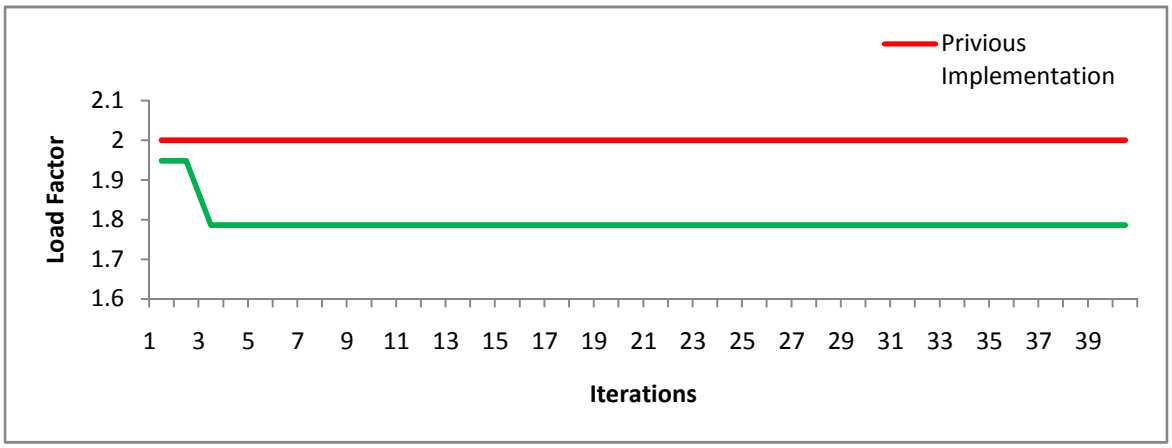

Figure 21. Comparison between load factors computed using both algorithms for the frame of example 4

7. CONCLUSION

In this work, two variants of ACS algorithm are proposed for the plastic analysis of planar frames. By eliminating redundant and time consuming parts in the first variant most of the computation time devoted to entangled ants is saved in an efficient way. Specializing the meaning of distance in classical ant colony optimization for the problem at hand enables to remove unnecessary steps to compute an acceptable distance parameter required for the operation of ACS algorithm. For certain configurations, these modifications lead to a considerable decrease in computational effort while preserving the accuracy and stability of algorithm. For special combinations of plastic moments and loadings, the provisions employed in the first variant do not lead to acceptable results. Therefore a second variant is proposed which alleviates the constraints in the first variant widening the range of frame configurations to which the algorithm can be applied at the cost of higher computation time. Several numerical examples are presented with small to medium scale which demonstrates the efficiency and performance of proposed algorithms. It is shown that the algorithm can compute the collapse load factor for frames encountered in practical applications with sufficient accuracy and reasonable computation time.

REFERENCES

[1] J. Baker, MR. Horne, J. Heyman, *The steel skeleton plastic behavior and design*, Cambridge University Press, Cambridge, UK, 1956.
[2] BG. Neal, PS. Symonds, The rapid calculation of plastic collapse loads for a framed structure, *Proceedings of the institution of civil engineers*, London, 1952, pp. 58–100.
[3] BG. Neal, PS. Symonds, The calculation of plastic loads for plane frames, *International association for bridge and structural engineering, fourth congress*, Cambridge, 1952.
[4] BG. Neal, PS. Symonds, The calculations of collapse loads for framed structures, *Journal of the Institution of Civil Engineers*, Vol.35 (1951) 21–40.
[5] A. Charnes, HJ. Greenberg, Plastic collapse and linear programming, *Summer Meeting of the American Mathematical Society*, 1959.
[6] J. Heyman, On the minimum weight design of a simple portal frame, *International Journal of Mechanical Sciences*, Vol.1 (1960) 121–34.
[7] MR. Horne, Determination of the shape of fixed ended beams for maximum economy according to the plastic theory, *International association of bridge and structural engineering, fourth congress*, Cambridge, 1953.
[8] J. Baker, J. Heyman, *Plastic design of frames, fundamentals*, Cambridge University Press, Cambridge, UK, 1969.
[9] A. Jennings, Adapting the simplex method to plastic design, *Proceedings of instability and plastic collapse of steel structures*, 1983, pp. 164–73.
[10] VB. Watwood, Mechanism generation for limit analysis of frames, *Journal of the Engineering Mechanics Division ASCE;* Vol.109(1979):1–15.
[11] MR. Gorman, Automated generation for limit analysis of frames, *Proceedings of ASCE ST7*, (1981).
[12] G.Thierauf, A method for optimal limit design of structures with alternative loads, *Computer Methods in Applied Mechanics and Engineering*,;Vol. 16 (1987) 134–49.
[13] A. Kaveh, *Optimal structural analysis*, John Wiley, 2nd edition, Chichester, UK, 2006.
[14] A. Kaveh, K. Khanlari, Collapse load factor of planar frames using a modified genetic algorithm, *Communications in Numerical Methods in Engineering*; 20 (2004) 911–25.
[15] J. Munro, Optimal plastic design of frames, *Proceedings of the NATO advanced study in engineering plasticity by mathematical programming*, Waterloo, Canada, (1977).
[16] RK. Livesley, Linear programming in structural analysis and design, *Proceedings of optimum structural design*, New York, USA, (1977).
[17] MJ. Best, Engineering plasticity by mathematical programming, *Proceedings of the NATO advanced study in engineering plasticity by mathematical programming*, Waterloo, Canada, (1977).
[18] G. Maier, J. Pastor, ARS. Ponter, D. Weichert, Direct methods in limit and shakedown analysis, Chapter 12 in: vol. 3, de Borst R, Mang HA, Eds., Numerical and computational methods, in: I. Milne, RO. Ritchie, B. Karihaloo, Eds., *Comprehensive structural integrity*, Elsevier-Pergamon, Amsterdam, 2003.
[19] A. Kaveh, M. Jahanshahi, Plastic design of frames using heuristic algorithms, *Proceedings of the eighth international conference on computational structures technology*, Stirlingshire, Scotland, 2006.
[20] S. Pellegrino, CR. Calladine, Structural computation of an assembly of rigid links, frictionless joints, and elastic springs. *Journal of Applied Mechanics ASME* 58 (1991) 749–53.
[21] AJ. Deeks, Automatic computation of plastic collapse loads for frames, *Computer and Structures* 1996; 60:91–102.
[22] A.Topc_u, *A contribution to the systematic analysis of finite element structures using the force method*, Doctoral dissertation, Essen University, 1979.
[23] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperative agents, IEEE Trans Syst Man Cyber Part B, 26 (1996) 1–13.
[24] M. Dorigo, T. Stützle *Ant colony optimization*, Prentice Hall, 2005.
[25] A. Kaveh, *Structural mechanics: graph and matrix methods*, Research Studies Press, 2004.
[26] A. Kaveh, M. Jahanshahi, Plastic limit analysis of frames using ant colony systems, *Computer and Structures*; 86 (2008) 1152-1163.
[27] A. Kaveh, M. Jahanshahi, M. Khanzadi, Plastic analysis of frames using genetic and ant colony algorithms, *Asian Journal of Civil Engineering*, Vol. 9 (2008) 229-249.
[28] A. Kaveh, M. Bakhshpoori, M. Kalateh-Ahani, Optimum plastic analysis of frames using ant colony system and charged system search algorithms, *Sci Iran Trans A* 2013; 20(3):414-421