ASIAN JOURNAL OF CIVIL ENGINEERING (BUILDING AND HOUSING) VOL. 9, NO. 3 (2008)
PAGES 229-249

PLASTIC ANALYSIS OF FRAMES USING GENETIC AND ANT COLONY ALGORITHMS

A. Kaveh*, M. Jahanshahi and M. Khanzadi
Department of Civil Engineering, Iran University of Science and Technology, Narmak, Theran-16, Iran

Abstract

In the recent years heuristic algorithms such as genetic algorithms, ant colony algorithms and simulated annealing have found many applications in optimization problems. The essence of these algorithms lies in the fact that they do not depend on the specific search space to which they are applied and consequently this extends their generality. In this paper, genetic and ant colony algorithms are used to find the collapse load factor of two-dimensional frames and their efficiency is compared to a direct approach. It is shown that when these algorithms are tuned finely and their parameters are adjusted carefully, very good results can be obtained. Four examples are presented to illustrate the efficiency of algorithms.

Keyword: Plastic load factor; analysis; design; genetic algorithm; ant colony algorithm; planar frames

1. Introduction

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

The problem of plastic analysis and design of frames with rigid joints has been solved in the form of a 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], Theirauf [12], and Kaveh [13] among others. Considerable progress has been made in past decades, a complete reference of which can be found in Munro [14] and Livesley [15]. Plastic analysis and design of frames using combination of elementary mechanisms has some limitations, which prevent its use as a common tool for analysis. Among such limitations, one may refer to the extensive numbers of mechanisms, which should be generated. The tedious work of combining these

* E-mail address of the corresponding author: alikaveh@iust.ac.ir (A. Kaveh)

mechanisms to find the true mechanism is another drawback associated with this method. There is also the possibility that the assumed collapse mechanism for a given frame and the corresponding loading is not the correct one and hence the computed collapse load factor will then be only an upper bound to the actual collapse load factor. Considering these problems it is important to accomplish an algorithm, which has the capability to of finding the collapse load factor and the corresponding mechanism as fast and accurate as possible.

Therefore two aspects of such an algorithm should be fastness and accuracy. Here we intend to provide some sort of compromise between these two issues. This is where the role of heuristic algorithms becomes more apparent.

In the recent years, heuristic algorithms such as genetic algorithm, ant colony and simulated annealing are extensively applied to various optimization problems. The essence of these algorithms lies in the fact that they do not depend on the specific search space to which they are applied and consequently this extends their generality. In the literature direct methods have been used to find collapse load factor of two dimensional frames (see for example Watwood [10] and Deeks [16]). For direct methods, elementary mechanisms are iteratively combined until the final mechanism with the least collapse load factor is obtained. It is however observed that when the configuration of a typical frame is rather complex direct methods fail to obtain the correct answer. This failure is mainly attributable to the fact that the combination of two elementary mechanisms might not decrease the load factor in preliminary stages and many combinations may be required so that active hinges are provided at proper locations. Developing an algorithm that takes this effect into account and finds the collapse load factor in a polynomial time is hard if not possible. The complexity that arises in developing a full proof direct algorithm motivates one to investigate the behavior of heuristic algorithms when they are applied to the problem of finding collapse load factor of a typical frame. Previous work in developing such algorithms can be found in the work of Kaveh and Khanlari [17], and Kaveh and Jahanshahi [18].

In this work, genetic and ant colony algorithms are used to find the collapse load factor of two-dimensional frames and their efficiency is compared with a direct approach. It is observed that if these algorithms are tuned finely and their parameters adjusted carefully, good results can be obtained. Four examples are presented to illustrate the efficiency of algorithms.

2. Generation of Elementary Mechanisms

In order to find a set of independent mechanisms, the method of Watwood [11] can be used. However, in this method joint mechanisms are also computed which is unnecessary because joint mechanisms can automatically be assigned to each joint. Axial deformations can also be neglected since mechanisms are the results of excessive deformations in rotational degrees of freedom leading to plastic hinges. Considering these assumptions, one comes up with a method similar to that of Pellegrino and Calladine [19] and Deeks [16].

In the present method, independent mechanisms are computed for an assembly of pin-jointed rigid bars. By rigid, it is specifically meant that no axial deformation exists, and bars can only have relative rotations. Consider a typical member as shown in Figure 1.

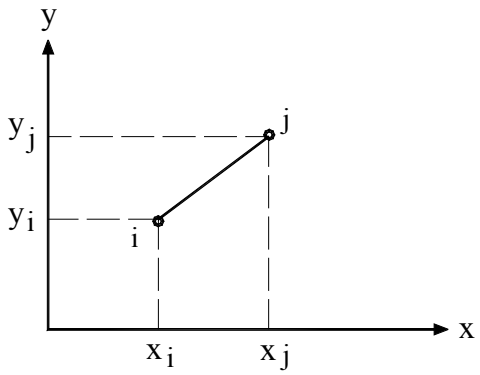

Figure 1. Configuration of a typical member

Specifying the elongation of each member in terms of its joint displacements, in the global coordinate system, leads to the following relationship.

$$e = (d_{xj} - d_{xi})\cos\alpha + (d_{yj} - d_{yi})\sin\alpha \tag{1}$$

Writing such expressions for all the members, the following equation is derived.

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

In a valid mechanism, members do not elongate. Therefore, in order to find the basic mechanisms, one should solve an equation for which elongation vector is zero, i.e. the following equation should be solved:

$$\mathbf{C}\mathbf{d} = \mathbf{0} \tag{3}$$

Since the assembly is not a truss and since it is not stable with all the joints being pinned, the number of columns of the coefficient matrix, **C**, exceeds the number of rows and the difference is the number of independent mechanisms (the dimension of null space of **C**). Performing Gaussian elimination on Eq. (3) results in the following form:

$$\left[\mathbf{I} \mid \mathbf{C}_d\right]\left\{\frac{\mathbf{d}^i}{\mathbf{d}^d}\right\} = \left\{\frac{\mathbf{0}}{\mathbf{0}}\right\} \tag{4}$$

In other words, the columns corresponding to independent displacements $\mathbf{d}^i$ are reduced to an identity matrix **I**, the order of which shows the dimension of the row space or the column space of C. Rearranging the terms in Eq. (4), the vector $\mathbf{d}^i$ can be expressed in terms of $\mathbf{d}^d$ as

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

Choosing dependent vectors for $\mathbf{d}^d$ with dimension as the number of independent mechanisms, and computing the independent vector $\mathbf{d}^i$ using Eq. (5), leads to the solution of Eq. (3). This results in independent mechanisms of the frame under consideration. As a simple computational approach, the dependent vectors can be constructed each time by setting one of the dependent displacements to unity and the remaining displacements to zero. The details of such an approach can be found in [16].

Though the mechanisms obtained in this way can be used in the linear programming methods, however, these mechanisms are not suitable for the method of combining mechanisms. They contain more active hinges than it is necessary for a logical collapse mechanism. An acceptable collapse mechanism can be obtained by just removing one active hinge. The mechanisms obtained using the above described method may contain all the active hinges of another mechanism as well.

Following the method of Deeks [16], independent mechanisms can be purified by removing the excess hinges in order to obtain a set of potential collapse mechanisms. This is achieved by checking each independent mechanism for containing a complete set of active hinges of another independent mechanism. For such case purification is performed by removing the contained mechanism. This process is repeated over and over until no modifications can be made.

3. Determination of Collapse Load Factor

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

$$\lambda_c = \frac{\text{internal virtual work}}{\text{external virtual work}} \tag{6}$$

The external virtual work is computed by adding up the products of all joint forces **P**, and the corresponding joint displacements **d** in the direction of these forces.

$$\text{External virtual work} = \mathbf{P}^t\mathbf{d} \tag{7}$$

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

$$\text{Internal virtual work} = \mathbf{M}^t|\mathbf{r}| \tag{8}$$

Since the joint mechanisms have been neglected during the formation of independent

mechanisms, it is necessary to find the location of hinges in the members. These locations are determined to minimize the internal virtual work.

If a joint is restrained against rotation, hinges are formed in all the members connected to that joint. However, if the joint is not restrained against rotation, hinges are formed in n−1 members among n members connected to that joint. In this case, n possible locations for hinges exist and it is necessary to find a location which minimizes the internal virtual work. When the number of hinges formed is less than the maximum number of hinges, the rotation in one or more of the assumed hinges is zero and does not contribute to the virtual work, Deeks [16].

4. Combination of Elementary Mechanisms

After generating the elementary mechanisms, it is necessary to combine suitable mechanisms to obtain a logical collapse mechanism with the lowest load factor. An experienced analyzer would choose the correct mechanisms and derive the true load factor in a short time. However, this intuition does not work in a computer oriented algorithm, since it is desired to keep the human interference to a minimum. Hence, it is necessary to consider all possible combinations of elementary mechanisms. Some mechanisms may increase the load factor when combined by the current mechanism but they may decrease it after several steps has been accomplished. Therefore, when a mechanism does not reduce the load factor, this does not necessarily mean that it should be excluded from the process of combination. The criterion for a mechanism to be the correct collapse mechanism is that there is no possibility for combining it with other elementary mechanisms without increasing the load factor.

Following the method of Neal and Symonds [2], an algorithm can be designed to check every possible combination by starting with an elementary mechanism and combining that mechanism with other elementary mechanisms in order to reduce the load factor. When no more reduction can be achieved by combining other elementary mechanisms with the current one, the same process is repeated for the next elementary mechanism. When all elementary mechanisms are exhausted, it is certain that no possibility is left and the correct collapse load factor cannot further be reduced.

Some authors use recursive functions to combine the mechanisms. Recursion has the draw back that it might shut down by blowing up the system stack if it does not have enough space to accommodate the address of calling recursive function. This can be remedied by using an endless loop instead of recursion and a good condition for jumping out of the loop when all possibilities are checked.

5. Genetic Algorithms

When the number of bays and stories of a typical frame increase, the number of mechanisms to be combined also raises. Then finding the correct collapse mechanism becomes a formidable task and requires a great deal of computational time.

It would be advantageous to find a good approximation to the actual collapse load factor

in a short time rather than a correct load factor in a long time. In this section, genetic algorithms are presented as an alternative to the method of combination of elementary mechanisms discussed in the previous section.

Genetic algorithms as the name suggests are good simulations of natural reproduction. Genes with better fitness have higher probability to survive and mate with other survivors to reproduce new generations. Samples that are produced from generation to generation have the properties that they inherit good aspects of their parents and eliminate weaker ones. We adopt these algorithms in order to choose appropriate elementary mechanisms to be used in our combination process. For this purpose, some definitions are necessary which are presented in the following.

Chromosomes are strings of binary bits, the number of which is taken as the number of independent mechanisms. A unit value for a bit means that the corresponding mechanism takes part in combination, and zero value it does not.

Mechanism:	1	2	3	4	5	6	7	8
Chromosome:	1	0	1	1	1	0	1	1

In the previous example, mechanisms 1, 3, 4, 5, 7 and 8 take part in combination process and mechanisms 2 and 6 do not.

Crossover is an operation in which two strings are crossed and new strings are generated. A crossing site should be selected with uniform probability P_c between the first bit and the last bit of the strings to which crossover operation is to be applied. The bits extending from the crossing site to the end of the string are exchanged. The following example shows how a crossover between two strings is performed.

A_1:	0	1	1	0	1
A_2:	1	1	0	1	0

Crossing site

A crossing site has been selected between bits 3 and 4. After crossover, the following strings are produced.

A'_1:	0	1	1	1	0
A'_2:	1	1	0	0	1

Mutation is the random change of a randomly selected bit from 1 to 0 or vice versa.

In order to begin the search for the lowest load factor, an initial generation is produced randomly and Genetic operations are performed on it. Fittest individuals are copied to a new generation and the same process is repeated over and over until a fairly good approximation

is obtained. The measure for fitness is the result of evaluation of a specific function called fitness function defined as follows.

$$f_i = C - \lambda_i \tag{9}$$

In which f_i, λ_i and C are the fitness function for chromosome i, the corresponding load factor, and the maximum load factor in the current generation, respectively. Thus the problem of minimizing the load factor is transformed into the maximization of the fitness function.

In our experiments, the population size and the number of generations have been set to 100 and 50, respectively.

6. Ant Colony Algorithms

Ant Colony algorithms are another type of optimization algorithms that are used to find the final failure mechanism of a typical frame. The building blocks of these algorithms are cooperative agents called "*ants*" [20]. These agents encompass simple capabilities, which make them resemble the behavior of real ants. It is known that ants communicate information through the pheromone trails. Ants lay pheromone on the ground and in this way they mark the path, which leads to the source of food. Although individual ants may move in quite an arbitrary direction, but they can detect a previously laid trail of pheromone and when such a trail is found, there exists a great probability for them to follow it. The final result is that more ants tend to pass through the path and the path becomes more important as the amount pheromone being laid 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 A to the food source F, and vise versa (paths ABCEF and ABDEF). Also assume that there are 30 ants deciding to move from A to F. At first there exists equal probability for ants to select the either path, so at point B roughly 15 ants select BCE and the other 15 select BDE. Since the path BCE is shorter than BDE, ants selecting this path reach E sooner than the others. The result is that the ant returning back from F to A finds more pheromone trail laid on BCE either by half of the ants that selected BCE by chance and by those that have already got back to A via BCE. Consequently the number of ants selecting BCE increases as the amount of pheromone laid on this path increases with time.

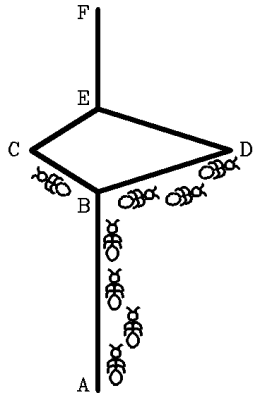

Figure 2. Paths from nest A to food source F

To get more insight into the problem suppose that the distances from B to C and C to E are equal to 0.5 and those of B to D and D to E are equal to 1. We are concerned to know what happens at discrete time steps t=1, 2… Assume that at t=0, 30 ants come to B from A and 30 comes to E from F. Also assume that each ant walks at speed of 1 per time unit and lays down a pheromone trail of intensity 1 that evaporates completely after each time step.

At t=0 there is no trail on the paths from B to E. The probability of choosing either of the two paths is equal and thus moderately 15 ants move towards C and 15 towards D.

At t=1 the new 30 ants that come from A to B find a trail intensity of 15 on the path leading to D laid by the 15 ants that went from B to D and an intensity of 30 on the path leading to C laid by the other 15 ants that went from B to C and 15 ants that came from E to B via C. The probability of choosing the path from B to C is therefore doubled according to the amount of pheromone laid on BCE. This process is continued until all of the ants will eventually choose the shortest path.

The idea is that if an ant is to choose among different paths, those with higher level of pheromone intensity are more likely to be chosen. Furthermore, a high level of pheromone intensity is equivalent to a shorter path.

In the literature the ant colony algorithms are usually introduced with the help of Traveling Salesman Problem (TSP). Here we also give a brief description of TSP to complement the description of the ant colony algorithm. The interested reader is encouraged to consult reference [20] for more details.

Given a set of n towns, the TSP is the problem of finding a closed tour of minimum length in which every town is visited exactly once. The distance between towns i and j is called d_{ij}, which in Euclidean space is the real distance between those towns. The TSP problem can be represented as a graph consisting of the set of nodes N representing the n towns and the set of edges E representing the paths between towns.

Let $b_i(t)$ (i=1, 2… n) be the number of ants in the town i at the time t, and let m be the total number of ants. Each ant is a simple agent with the following capabilities:

- It chooses the town to move to with a probability that is a function of the town distance and the amount of trail present on the connecting edge
- To force the ant to make legal tours, transition to already visited towns are prohibited until a tour is completed

- When it completes the tour it lays a predefined amount of trail on visited edges

Assume that at time t, τ_{ij} is the intensity of trail on edge (i, j). At this time every ant chooses the town to move to at time t+1. Therefore in time interval (t, t+1), which is called an iteration of ant colony algorithm, m moves are carried out by the m ants. After performing n iterations or one cycle each ant completes a tour. At this point the trail intensity is updated according to the following formula:

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} \tag{10}$$

Where ρ is a coefficient such that (1-ρ) represents the evaporation of trail between time t and t + n, and

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{11}$$

Where $\Delta\tau_{ij}^{k}$ is the quantity of trail substance per unit length laid on edge (i, j) by the kth ant between the time t and t + n. The value of $\Delta\tau_{ij}^{k}$ is given by

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if kth ant uses edge (i, j) in its tour between time t and t + n} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Where Q is a constant, and L_k is the length of the tour for kth ant.

The coefficient ρ must be set to a value smaller than one to avoid unlimited accumulation of trail. At time t = 0, a small positive constant c is assigned to $\tau_{ij}(0)$.

Another important parameter used in ant colony algorithms is visibility which is defined as $\eta_{ij}=1/d_{ij}$ and unlike trail intensity is not modified and remains constant through iterations.

With the help of previous parameters, the probability of transition from town i to town j can be defined as:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^{\alpha}.[\eta_{ik}]^{\beta}} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Where the allowed_k is the set of towns among which the ant can choose to move to and α and β are parameters that control the relative importance of trail versus visibility. Therefore the transition probability is a trade off between visibility and trail intensity at a given time.

Given the basics of ant colony algorithms, it can now be adopted for computing the collapse load factor of frames. Assume that a typical frame has n elementary mechanisms. Initially a complete graph consisting of n nodes representing the n elementary mechanisms

is constructed and 2 ants are placed on each node. Ants have the extra capability of saving the mechanism resulting from combination of two elementary or combined mechanisms. Ants move from one node to another node based on pseudorandom proportional rule. In each movement they save the mechanism resulting from the combination of mechanisms that they already saved with the mechanism corresponding to the node to which they are bound to. Based on this logic, the meaning of distance is slightly different from that of the TSP. Consider the node i on which the ant is placed and a node j to which the ant has chosen to move to, and finally the mechanism resulting from the combination of ant's mechanism (the combined mechanism saved by ant up to node i) with mechanism corresponding to node j. Denoting the load factors of ant's mechanism and mechanism corresponding to node j as F_i and F_j and that of the combined mechanism F_c, the distance from node i to node j can be computed as follows:

$$d_{ij} = F_{max} + F_c - F_i \tag{14}$$

Where F_{max} denotes the maximum load factor of the elementary mechanisms. The idea is that when an ant is placed on a typical node i, it chooses to move to another node j. If the combination of mechanism already saved by ant with the mechanism corresponding to node j results in a smaller load factor, the process of combining takes place and ant moves from node i to node j otherwise it is skipped.

In order to diversify the search space, the best combined mechanism obtained by ants in each cycle is added to the set of elementary mechanisms. In other words, the initial complete graph consisting of n nodes is expanded to a complete graph consisting of n + 1, n + 2… nodes. The size of expansion in this work is chosen to be limited to 2n nodes although other alternatives could be considered as well. After the limit is reached, new mechanisms replace old inferior ones. However, as the elementary mechanisms comprise the essence of combination, a differentiation between the original n nodes (corresponding to the elementary mechanisms) and the new n nodes (corresponding to the combined mechanisms) has to be made. This means that in subsequent cycles new combined mechanisms replace combined mechanisms with higher load factors and elementary mechanisms remain intact. The concept of expansion is shown in Figure 3, where initially the graph consists of n nodes corresponding to n elementary mechanisms (Figure 3a) and after the first cycle it expands to a graph consisting of n + 1 nodes, where n nodes corresponds to elementary mechanisms and the new node corresponds to new combined mechanism. Other free nodes in Figure 3b imply that the graph ultimately expands to 2n nodes. To emphasize that the original n nodes remain intact in replacement process, two parts of the graph corresponding to original nodes and new nodes are isolated by drawing ellipses around them. After expansion, new cycles start with ants placed on new nodes as well.

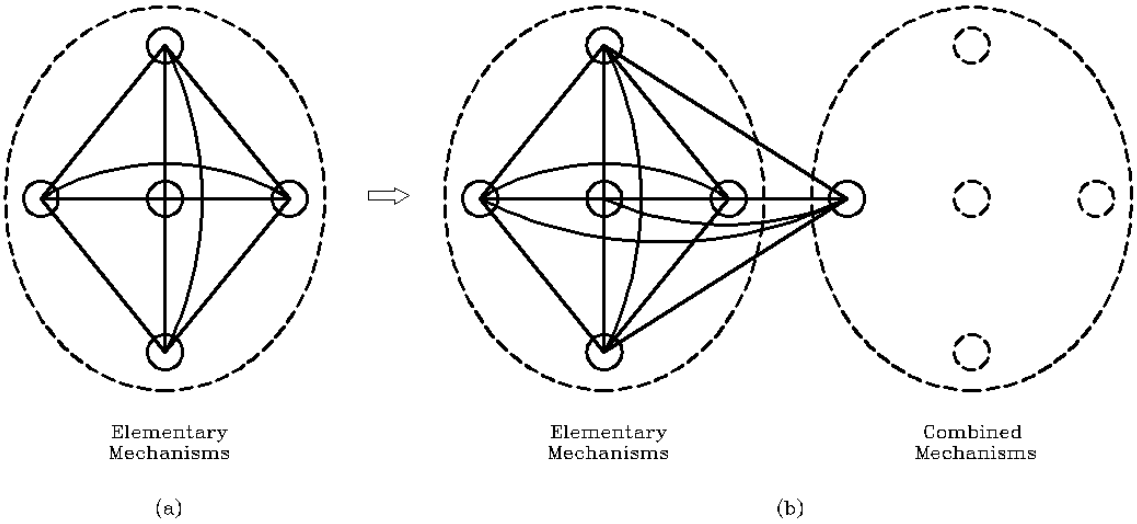

Figure 3. The process of expansion: (a) Original graph, (b) Expanded graph.

An important issue which has to be taken into account is the initialization and updating the pheromone matrix when the graph is expanded or when a new node replaces an old one. Initialization is easy to perform since one can initialize the matrix corresponding to 2n nodes (instead of n nodes) at the beginning of algorithm. However, replacing nodes produces many problems because one should keep the track of those elements of the pheromone matrix that are affected during pheromone updating by ants traveling from other nodes to the replaced nodes or vice versa. Besides, the process through which the values corresponding to new nodes are updated is also questionable. In this work a simple scheme is chosen for updating. As the new nodes which replace the old nodes with higher load factors are more favorable, and it is desirable to motivate ants to move towards these new nodes, it is sufficient to add to the corresponding rows and columns a value equal to the difference between the replacing and the replaced load factors. In this way the probability of choosing new nodes in subsequent cycles is increased.

In our experiments, 1.0, 5.0 and 0.5 are used for α, β and ρ, respectively. The number of cycles was set to 20. Pheromone matrix is initialized with a small positive value. It is observed that lowering the β value to 1.0 while keeping α constant does not affect the final result. The reason for this insensitivity to β parameter is that a local search is also working in addition to ant colony meta-heuristic. It should be remembered that when a node j was selected as a result of pseudorandom proportional rule, transition to that node was only allowed when the final load factor was smaller. However, increasing α while keeping β constant misleads the algorithm and higher load factors are obtained. As an example, choosing 2.0 for α and 1.0 for β leads to a load factor of 0.65041 for Example 4 of the following section.

At the end, the procedure followed by ants to find the minimum collapse load factor is described using a high level pseudo code shown in Figure 4. In this figure, it is observed that any initialization takes place in InitializeData. Afterwards the main loop starts in which ants

are assigned to mechanisms. Initially only elementary mechanisms are present and since the number of ants is twice the number of elementary mechanisms, two ants are assigned to each mechanism. However when the expansion process is complete each ant is assigned exactly to one mechanism. In ConstructSolution ants move from node to node based on a decision rule building a partial combined mechanism. Each ant builds a tour in which the number of visited nodes is at most equal to the number of existing mechanisms minus one (skipping the movement back to initial node). In each iteration, after all ants have built their respective tours, the pheromone matrix is updated and the best mechanism is saved. This mechanism gets added to elementary mechanisms and thus expands the graph or it replaces the worst combined mechanism if the expansion process is complete. Finally ants die and a new iteration starts.

```
procedure FindCollapseMechanism
    InitializeData
    while (termination condition not met) do
        PositionAntsOnIndividualMechanisms
        ConstructSolution
        UpdatePheromoneMatrix
        SaveTheBestCombinedMechanism
        RemoveAnts
    end while
end procedure

procedure InitializeData
    InitializeAnts
    InitializePheromoneMatrix
end procedure

procedure ConstructSolution
    for i = 1 to number_of_iterations
        for j = 1 to number_of_mechanisms
            for k = 1 to (number_of_mechanisms - 1)
                MoveAntBasedOnDecisionRule(j)
            end for
        end for
    end for
end procedure
```

Figure 4. High level pseudo code describing ACO

7. Numerical Results

In this section, four examples are presented to show the performance of algorithms and to

provide a measure for comparing the results. These examples vary from simple one-bay frames to multi-story, multi-bay frames and it is intended to qualify the algorithms against different level of complexity. The computer time are also provided which present the CPU time consumed by each of the heuristic algorithms; genetic and ant colony. The computational time are computed using a 2.8GHz Pentium 4 computer equipped with a 1GB RAM.

Example 1: A gable frame is considered with the configuration and loading shown in Figure 5. The frame has four independent mechanisms and the collapse load factor obtained from the direct method, genetic and ant colony algorithms is 0.8182. Failure mechanism for all three methods is identical as shown in Figure 6. The CPU time for this example is provided in Table 1.

Table 1. CPU time for Example 1

Method	Time (seconds)
Genetic Algorithm	0.047
Ant Colony Algorithm	0.047

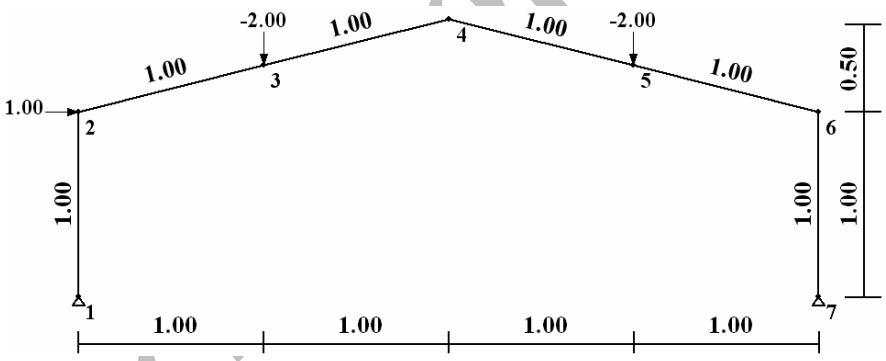

Figure 5. Gable frame, loading and plastic moments

Figure 6. Collapse mechanism

Example 2: Second example is a three-story frame as illustrated in Figure 7. The actual collapse load factor computed by the direct method is 1.97 (Figure 8) and that by genetic and ant colony algorithms is 2 (Figure 9). The mechanism corresponding to load factor 2 is a

simple beam mechanism while the mechanism corresponding to the load factor 1.97 affects the entire structure. The CPU time for this example is given in Table 2.

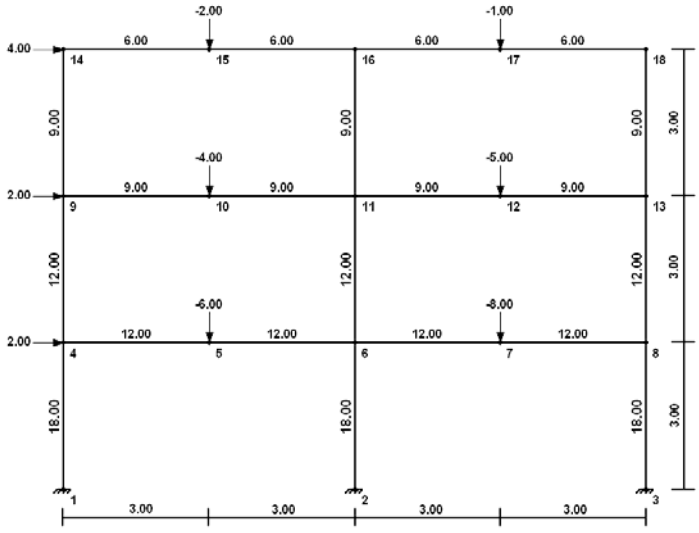

Figure 7. Two-bay, three-story frame, loading and plastic moments

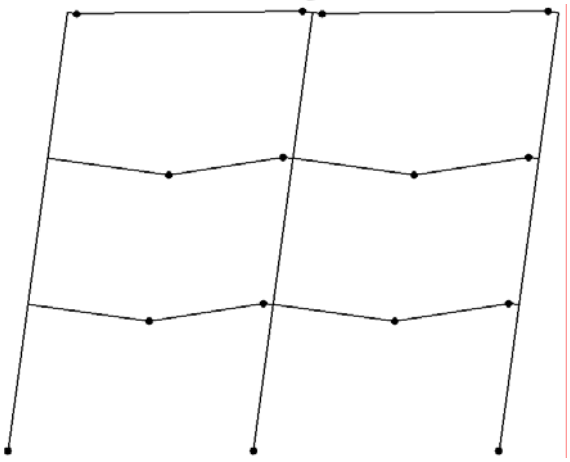

Figure 8. Collapse mechanism obtained from direct method

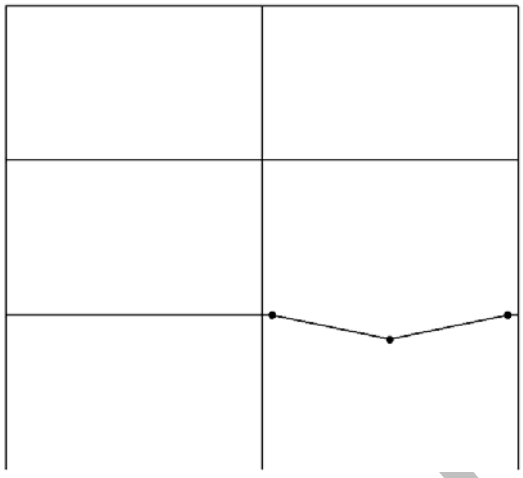

Figure 9. Collapse mechanism obtained from genetic and ant colony algorithms

Table 2. CPU time for Example 2

Method	Time (second)
Genetic Algorithm	0.219
Ant Colony Algorithm	1.297

The load factor corresponding to the beam mechanism is very close to the actual load factor and therefore makes it hard for the algorithms to find the correct load factor and collapse mechanism.

Example 3: A six-story frame is considered with the geometry and loading shown in Figure 10. The magnitude of the collapse load factor is 1.29 and is the same for all three methods. The failure mechanism is illustrated in Figure 11. The CPU time for this example is given in Table 3.

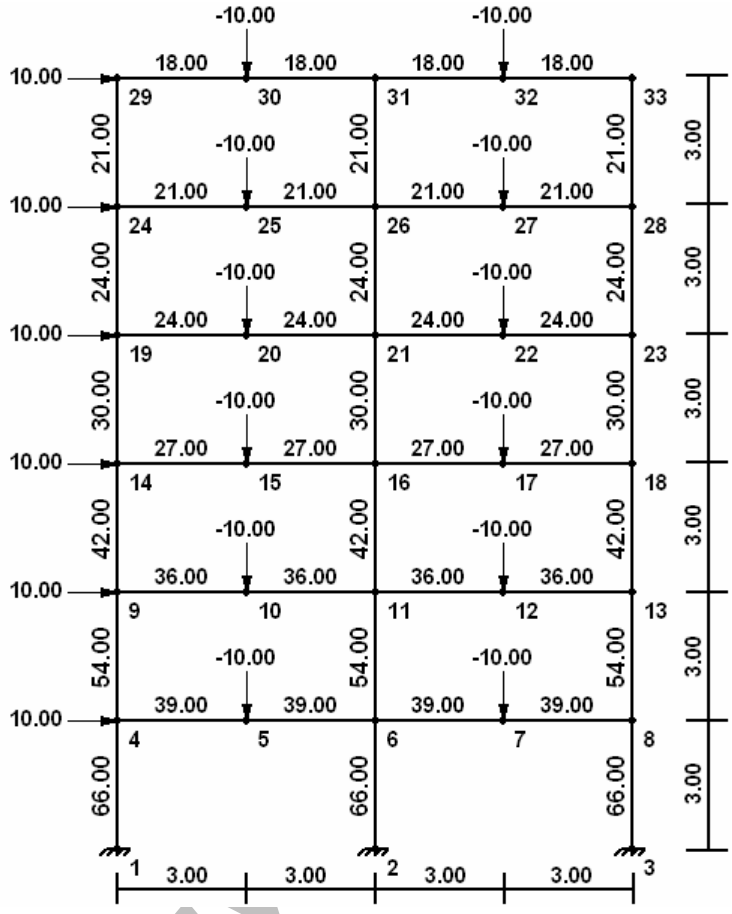

Figure 10. Two-bay, six-story frame, loading and plastic moments

Table 3. CPU time for Example 3

Method	Time (seconds)
Genetic Algorithm	0.797
Ant Colony Algorithm	17.344

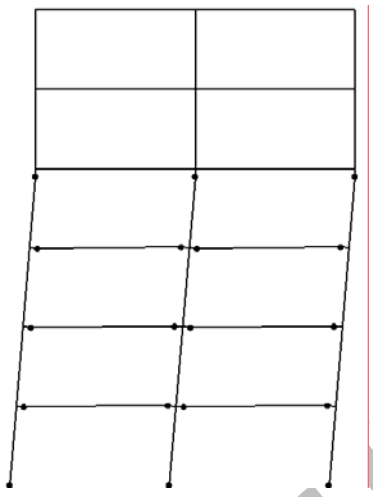

Figure 11. Collapse mechanism

Example 4: The fourth example is a four-bay, four-story frame that is provided as a more general structure to test the efficiency of algorithms. The frame configuration and its loading are shown in Figure 12. The actual collapse mechanism obtained by both the direct method and the ant colony algorithm and that by the genetic algorithm are shown in Figs. 13 and 14. The corresponding load factors are 0.65 and 0.67, respectively. The CPU time for this example is given in Table 4.

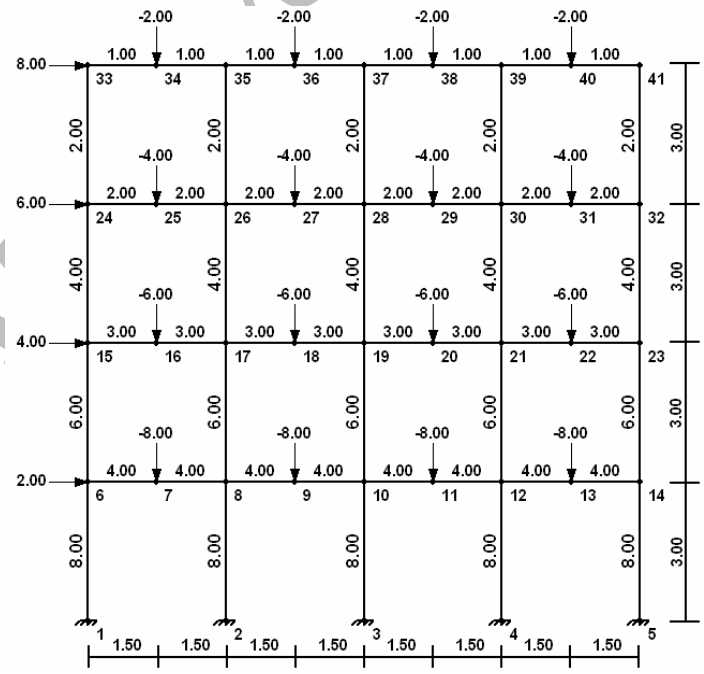

Figure 12. Four-bay, four-story frame, loading and plastic moments

Table 4. CPU time for Example 4

Method	Time (seconds)
Genetic Algorithm	0.954
Ant Colony Algorithm	31.969

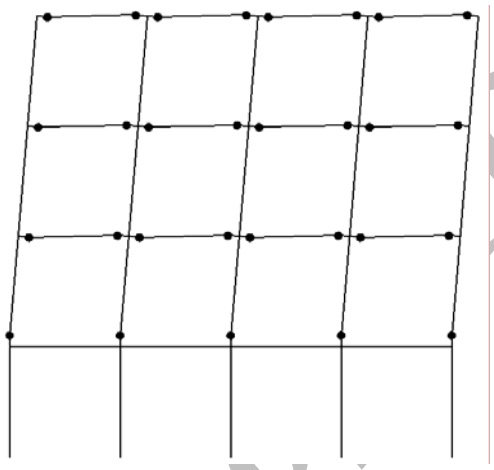

Figure 13. Actual collapse mechanism obtained by the direct method and ant colony algorithm

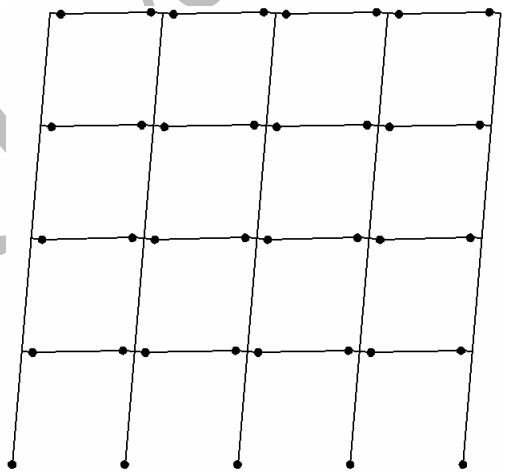

Figure 14. Collapse mechanism obtained by the genetic algorithm

8. Concluding Remarks

It is observed that both the genetic and ant colony algorithms compute reasonable load

factors for frame examples presented in this paper. However, the collapse mechanism might be different from the actual one. The difference is more pronounced for genetic algorithms but mechanisms found by ant colony algorithms are in many situations actual mechanisms or very close to it.

Time is another important factor in comparing genetic algorithms with ant colony algorithms. It is observed that for the parameters chosen (50 generations for genetic algorithm and 20 cycles for ant colony algorithm) genetic algorithm accomplishes the task of finding collapse mechanism much faster than that of the ant colony algorithm. This difference in time motivates one to investigate the convergence behavior of genetic algorithms versus ant colony algorithms.

The variation of the collapse load factor computed in each generation and cycle for the frame of Example 4 by the genetic and ant colony algorithms are shown in Figure 15 and Figure 16, respectively. The load factors that genetic algorithms compute have considerable oscillations and one should choose the minimum load factor ever found through the optimization process. This clearly means that it is not guaranteed that genetic algorithms will finally converge to the minimum load factor. However, the graph in Figure 15 shows that unlike genetic algorithms, ant colony algorithms converge to the final collapse load factor after a few cycles. For instance, at most 4 or 5 cycles are needed to compute the collapse load factor for the frame in Example 4. This is a considerable reduction compared to 20 cycles and it is suggested that one chooses a limited number of cycles to compute the collapse load factor and then increases the cycles to verify the results. Thus as a concluding remark it can be stated that if a reasonable number of cycles is chosen for ant colony algorithms, the time elapse compares well with that of genetic algorithm.

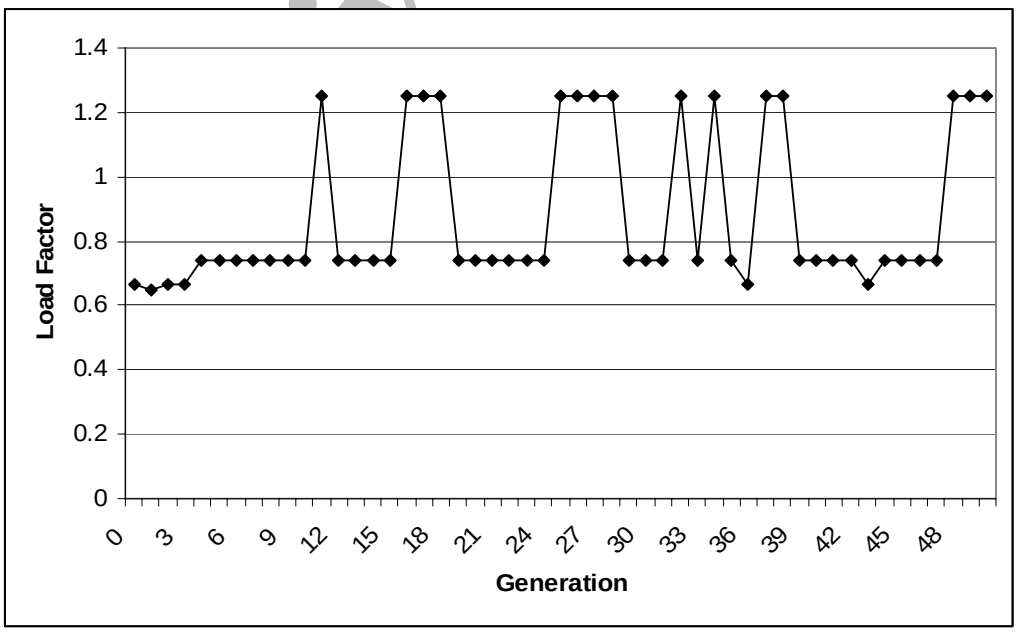

Figure 15. Load factor obtained by GA in each generation.

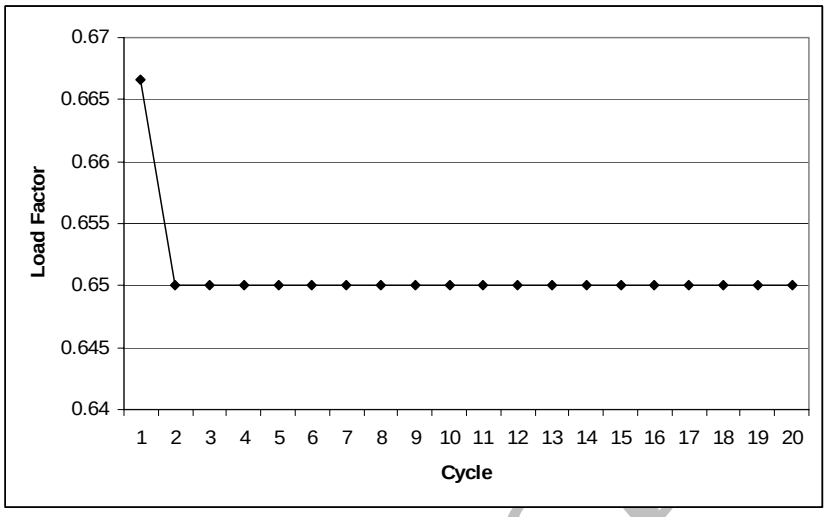

Figure 16. Load factor obtained by ACO in each cycle.

There is also a performance point about ant colony algorithms, which is noteworthy to be mentioned. New suboptimal mechanisms obtained in each cycle expand the graph up to twice the original size. This limit was arbitrarily chosen based on such factors as CPU time, memory accessibility and speed. The goal was to find out how the expansion idea in a predefined scheme (like the one chosen here) can affect the overall behavior of ant colony algorithms. No elaboration has been focused on relating the expansion size on the convergence route of algorithm. Based on this hypothesis, one can impose other constraints on the size of the graph and choose an optimal limit or dynamically modify it. It is certain that if the size is three, four… times the original size, the space in which the actual mechanism is searched for becoming larger and there is a strong probability for actual mechanism to reside in this space. However, it is important to pay attention to the time being consumed for searching such a large space. The compromise problem between time and speed again manifests itself.

Based on these observations it can be suggested that ant colony algorithms should be employed instead of genetic algorithms especially in situations where a greater safety bound is required.

Acknowledgement: The first author is grateful to the Iran National Science Foundation for the support.

References

1. Baker J, Horne MR, Heyman J. The steel skeleton; plastic behavior and design. Cambridge at the University Press, Volume 2, 1956.
2. Neal BG, Symonds PS. The rapid calculation of plastic collapse loads for a framed structure. Proceedings of the Institution of Civil Engineers, London, Part 3, **1**(1952)58-

100.
3. Neal BG, Symonds PS. *The calculation of plastic loads for plane frames*. Preliminary Publication, International Association for Bridge and Structural Engineering, 4th Congress, Cambridge and London, 1952.
4. Neal BG, Symonds PS. The calculations of collapse loads for framed structures, *Journal of Institute for Civil Engineers*, **35**(1951)21-40.
5. Charnes A, Greenberg HJ. Plastic collapse and linear programming. Summer Meeting of the American Mathematical Society, 1959.
6. Heyman J. On the minimum weight design of a simple portal frame. *International Journal of Mechanical Science* **1**(1960)121-34.
7. Horne MR. Determination of the shape of fixed ended beams for maximum economy according to the plastic theory. Final Report, International Association of Bridge and Structural Engineering, 4th Congress: Cambridge London, 1953.
8. Baker J, Heyman J. *Plastic Design of Frames, Fundamentals*. Vol. 1, Cambridge, Cambridge University Press, 1969.
9. Jennings A. Adapting the simplex method to plastic design. In: Marris, LJ, editor. Proceedings of Instability and Plastic Collapse of Steel Structures, 1983, pp. 164-173.
10. Watwood VB. Mechanism generation for limit analysis of frames. *Journal of Structural Division, ASCE* ST1, **109**(1979)1-15.
11. Gorman MR. Automated generation for limit analysis of frames. *Journal of Structural Division ASCE* ST7, **107**(1981)1350–54
12. Thierauf G. A method for optimal limit design of structures with alternative loads, *Computer Methods in Applied Mechanics in Engineering*, **16**(1987)134-49.
13. Kaveh A. *Optimal Structural Analysis*. John Wiley, First edition, UK, 1997.
14. Munro J. Optimal plastic design of frames In Proceedings of NATO, Waterloo: Advanced study in Engineering Plasticity by Mathematical Programming, 1977, pp.136-71.
15. Livesley RK. *Linear programming in structural analysis and design*. In: Gallagher RH et al editors, Optimum structural design, Chapter 6, New York; Wiley, 1977.
16. Deeks AJ. Automatic computation of plastic collapse loads for frames, *Computers and Structures*, No. 3, **60**(1996)91-102.
17. Kaveh A, Khanlari K. Collapse load factor of planar frames using a modified Genetic algorithm, *Communications in Numerical Methods in Engineering*, **20**(2004) 911-25.
18. Kaveh A, Jahanshahi M. Plastic limit analysis of frames using ant colony systems, *Computers and Structures*, **68**(2008)1152-63.
19. Pellegrino S, Calladine CR. Structural computation of an assembly of rigid links, frictionless joints, and elastic springs. *Journal of Applied Mechanics ASME* **58**(1991)749-53.
20. Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperative agents. *IEEE Transactions on Systems, Man, and Cybernetics*-Part B, No. 1, **26**(1996)1-13.