Hybrid genetic algorithm for the maximum clique problem combining sharing and migration

Roger Ouch, Kristopher W. Reese, Roman V. Yampolskiy
Computer Engineering and Computer Science,
University of Louisville
Louisville, Kentucky 40292

Abstract

The Maximum Clique Problem (MCP) has been studied for decades and is well known in graph theory as a problem that is difficult as it as it is known to be NP-complete. The MCP has a vast domain of application such as finance, biochemistry, bioinformatics, and many more. Many niching methods have been successfully applied in Genetic Algorithms (GA) to diversify the population and avoid getting trapped within local optima. In this paper, we propose an approach using the Sharing method and a Hybrid Genetic Algorithm (HGA) for the maximum clique problem. We also propose a non-evolutionary approach using a migration mechanism to boost the current HGA.

I. Introduction

A. The Maximum Clique Problem

The maximum clique problem is a classically studied combinatorial optimization problem, which comes from graph theory and has many domains of application. The maximum clique problem can be briefly summarized using the following definition.

Let's G= (V,E) be an undirected graph on N vertices, where V is the vertex set and E is the edge set. A clique in G is a subgraph of G in which there is an edge between any two vertices. The size of a clique is the number of vertices in the clique, and the maximum clique problem is to find the largest clique in a given graph G.

B. Need For Improved Technique For Huge Networks

Over the last few years, the size of networks has increased rapidly and the amount of information from massive data sets has become a highly researched computational challenge.

Some examples of fast growing networks exist in social networks and bioinformatics. The use of traditional methods for analyzing and treating the large amounts of information has become problematic. The nature of the MCP (NP-complete) explains why is it more complicated to find the maximum clique when the number of vertice N increases dramatically. Traditional heuristic approaches for finding the maximum clique problem have become insufficient for this new challenge.

Cheng et al. [1] investigated finding the maximal clique within massive networks. They introduced the concept of an H* graph where the maximum clique is computed in a small part of a large graph recursively one at a time using external memory. Their experiments were done on large graph with 10 million vertices and 80 millions edges.

C. Description of Genetic Algorithms

A well-known metaheuristic in evolutionary algorithms is the genetic algorithm. The earliest work related to GAs dates back to 1954 when Nils and Barricelli published a paper on computer simulation of evolution [2]. Genetic algorithms have been formalized in later years and date back as early as 1975 when J.H Holland published his papers on the subject [3].

A GA is a combinatorial optimization technique that mimics the process of natural evolution. The algorithm does not rely on building the final solution based on local search, which distinguishes itself from traditional approaches. A subset of feasible solutions of a problem is encoded in chromosomes that represent individuals in a population. The evolutionary process is then applied through population selection, crossover, and mutation operations. Each individual is then evaluated using a fitness function and the best of the individuals are transmitted into the next generation. The following procedure shows a basic genetic algorithm:

```
GA procedure(){
        Initialize population();
        Evaluate population();
        While not (End_Condition){
                Select parent();
                Crossover();
                Mutation();
                Evaluate population();
        }
}
```

Figure 1 Genetic algorithm

Since the 1990s, significant research has been conducted on genetic algorithm, especially for finding the maximum clique.

II. Related Work

Early works on the maximum clique problem were focused on greedy approaches. One of the first significant improvements on the maximum clique was done by Bron-Kerbausch[4]. This algorithm uses a recursive backtracking procedure that augments the clique by one vertex at a time. Tomita, Tanaka and Takahashi [5] proved that with a worst-time complexity $O(3^{n/3})$, the Bron-Kerbaush algorithm was reported as one of the fastest algorithms for listing all possible maximum cliques.

In 1986, Robson improved this algorithm by adding backtracking techniques in combination with more complicated case analyses and dynamic programming. This however increased the space complexity of the MCP [6].

A. Genetic Algorithm Methods For Max Clique

Genetic algorithms have been successfully applied to many NP-hard problems in various domains [7-8]. GA has also been successfully used on graph problems, particularly on the graph-coloring problem [9]. On the MCP, the first approaches using GA had poor performance compared to other local search techniques [10, 11].

The idea of combining a genetic algorithm and a heuristic local search has been used in earlier applications. Marchiori [12] proposed a simple genetic algorithm based on a combination of heuristic algorithms for the MCP. In this HGA several important genetic mechanisms were implemented such as the keep-two-best parents, elitism, a roulette-wheel population selection, uniform crossover, and a fitness function based on the size of the maximal clique. Moreover, the heuristic method is used to do local transformation on a sub-graph transforming all chromosomes after crossover and mutation into a maximal clique. These processes have become known as the "relax", "repair", and "extend" phases. Her experiments on the DIMACS data sets showed that results quality and computational time have improved dramatically compared to other heuristic approaches. Marchiori [13] continued developing the hybrid genetic algorithm and compared this algorithm with two other competitive variations of genetic algorithm: the iterated local search algorithm and the multistart local search algorithm.

B. Fitness Sharing Method

Fitness sharing method has been used to avoid GA converging to local maxima. This technique was proposed by Holland [3] and expanded by Goldberg and Richardson [22]. In a GA, the formation of population in local optimal (also called niche) makes difficult the convergence toward the global optimal solution. The fitness sharing method helps to diversify the population by reducing the fitness score of individual in populated landscapes. This technique is known as one of the best niching technique to escape from local optima.

In this paper, we implemented the sharing method, as we are attempting to diversify and explore the elements of the graph more thoroughly. The sharing function recalculates the fitness function based on how dense an area of the graph is used. Therefore, as with any fitness function, our shared fitness becomes that shown in equation 1.

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{n} sh(d(i,j))} \qquad (1)$$

where sh is the sharing based on the distance d between vertex i and j. The sharing function is found in equation (2).

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha}, & if\ d < \sigma_{share} \\ 0, & otherwise \end{cases} \qquad (2)$$

C. Migration in GA

Yet another technique that inspired sociological and ecological movements between sub-populations is the concept of migration. Migration in genetic algorithms has been well studied and applied in many parallel genetic algorithms.

There are important considerations before implementing migration in genetic algorithms. Migration policies define the replacement rule to apply between two sub-populations. It is equivalent to answering the question: "What kind of individual in a sub-population A is replacing what kind of individual in sub-population B". Topology is another important aspect in multi-population genetic algorithm.

The topology defines the structure of communication between several sub-populations [14]. Finally, the migration rate must be used to define the proportion of a sub-population that is replaced with the immigrant individuals.

Migration rates and migration policies have been shown to improve the convergence of genetic algorithm considerably. Cantu-Paz has conducted experiments in which the best-replacing-worst population policy has demonstrated the best results for migration policies [15].

III. Proposed Solutions

A. Hybrid Genetic Algorithm

The heuristic local search algorithm is a local search based on greedy approach. Given a set of vertices, this algorithm finds the maximal clique using a "Relax", a "Repair" and an "Extend" steps. Marchiori described this algorithm in details [12]. The relax step adds random vertices to a subgraph. The repair step then extracts a small maximal clique from the current sub-graph. Finally the extend step generates a bigger maximal clique by adding random nodes to the clique obtained in the repair step.

The HGA is then obtained by combining the heuristic local search with a genetic algorithm given in figure 1. The heuristic function is added after the crossover operation to find maximal clique.

B. Introduction of a Migration Mechanism in Genetic Algorithm

In our solution we propose a migration mechanism based on two sub-populations, a main sub-population and a secondary sub-population. The main population will have the role of a standard HGA. The secondary sub-population will serve as a pool that generates the new chromosomes for migrating into the main subpopulation. This mechanism should help to accelerate the convergence of the current hybrid genetic algorithm.

1) Role of the Two Sub-Populations

The idea of combining a heuristic local search and a genetic algorithm is not new. It has proven to be competitive in solving the maximum clique problem. One issue with this approach is the convergence into local optima. The diversification strategy relies on the genetic operators: crossover and mutation. The exploration of the set of feasible solutions (or neighborhood) in hybrid genetic algorithm is an area that can be improved.

In our approach we define two sub-populations. The Main sub-population uses a generic HGA defined previously. This sub-population specializes in the exploitation of the data. The secondary sub-population will be dedicated to exploration of the data. In the secondary population, an effort is made to diversify the population. The sub-population will then transfer the best of its population to the main sub-population through a migration mechanism. In this approach, we want to maximize the functions of exploration and exploitation of the genetic algorithm. Figure 2 shows the role of each sub-population.

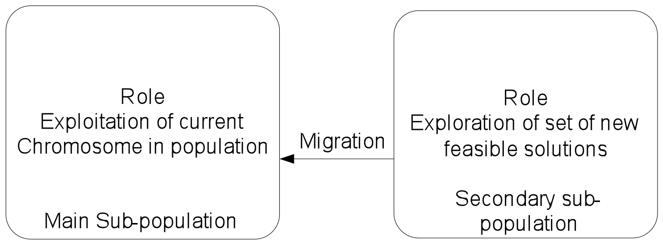
Role
Exploitation of current Chromosome in population
Main Sub-population
Migration
Role
Exploration of set of new feasible solutions
Secondary sub-population

Figure 2 Role of sub-populations

2) Migration Mechanism

In our algorithm, it was decided to use a best-replace-worst migration policy at each generation, where the best individuals in the sub-population will replace a percentage of the worst individuals in the main sub-population. We also used a ladder migration topology defined in [15], which allows the migration between the two sub-populations to occur over each generation of the system. Figure 3 shows the implementation of our hybrid genetic algorithm with a migration mechanism.

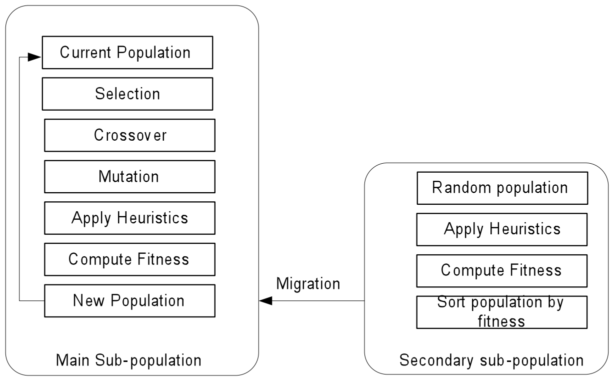

Figure 3 Hybrid GA with migration mechanism

The main sub-population is the standard genetic heuristic algorithm discussed by Marchiori [11]. The secondary sub-population, however, is of more interest. This subpopulation uses a non-evolutionary structure. It has no genetic operators and its role is to give random "seed" to the main population at every generation.

To initialize the population, we randomly choose vertices for each chromosome in the new population. We then apply the heuristic function, which extracts the maximal clique from each chromosome. We then sort the population by fitness, to determine which of the elements should migrate to the main sub-population.

IV. Experimental Results

A. Benchmark Datasets Used

The data used for our experiments are from the DIMACS (Center for Discrete Mathematics and Theoretical Computer Science) benchmark graphs. It is composed of a collection of 9 different classes of graphs for evaluating and comparing different algorithms for solving the maximum clique problem. This collection consists of random graphs with known maximum clique size as well as graphs obtained from various areas of applications. The C-FAT graphs are based on the fault diagnosis problem [16]. The Hamming and Johnson graphs are from the coding theory problems [17] [18]. The Keller graphs arise from the Keller conjecture on tiling using hypercube [19]. The SAN, SANR, BROCK and P-HAT are composed of various types of random graphs with known maximum clique sizes. The BROCK graphs contain random graphs constructed so that they have hidden cliques that are larger than what might be expected in a random graph. The P-HAT graphs are random graphs with large variance in the vertex degree distribution and a larger clique than usual random graphs [20]. Finally, the MANN graphs are from the vertex-covering problem, which is closely related to the maximum clique problem [21].

B. Results Obtained

A series of experiments were conducted using a small subset of the Dimacs data sets. The results obtained show that the performance of our algorithm is close to the best results obtained on the same data sets from the DIMACS challenge (table 1).

Graph	Best obtained	Best DIMACS
keller4	11	11
keller5	27	27
keller6	49	59
hamming8-4	16	16
hamming10-4	38	40
MANN_a27	126	126
MANN_a81	1096	1098

Table 1 Comparison of our algorithm with Best DIMACS

C. Effect of Mutation Rate Variation on Results

In these experiments, we evaluate the effect of the mutation by holding other parameters static. We analyzed the effect of mutation on the Keller5 graph with the mutation rate parameters: 0.5%, 1%, 5%, 10% and 20%. Table 2 shows the parameters chosen for this set of experiments.

Parameters	Value
Population size	100
Maximum iteration	50
Mutation	0.5%, 1%, 5%, 10%, 20%
Crossover rate	100% (uniform crossover)

Table 2 Parameters for testing mutation rate

At each generation, the mutation breaks current maximal clique in every chromosome, by replacing current nodes by other random nodes.

On one hand, a low mutation rate keeps nodes that are good candidates for the maximum clique and the heuristic algorithm find good local solutions. But less graph exploration are possible using a low mutation rate. On the other hand, a high mutation rate gives a good chance of graph exploration, but the local search become less efficient, as to many nodes are exchanged in chromosomes. Choosing a good mutation rate is a trade-off between finding a good local solution and good graph exploration. Figure 4 shows the minimum, the average and maximum fitness score for each chromosome for 1% mutation rate.

We can observe two phases applying the GA to the maximum clique problem. In the first phase, a high variation of the fitness score phase can be observed from 0 to the 15th generation. During this phase, the HGA explores various combinations of nodes in the graph, so the fitness score change rapidly. Then a phase of stabilization is observed where the maximal cliques are kept in the population and improvement occurs progressively. During the stabilization phase, the chances of discovering bigger cliques are low because the HGA reach local maxima.

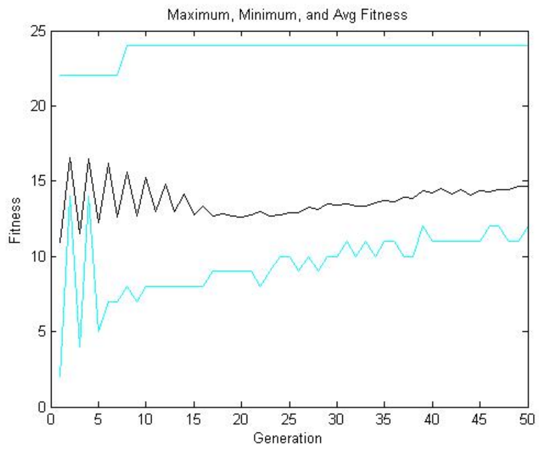

Figure 4 mutation= 1%

The results obtained in our experiments shows that 1% mutation rate achieves the best performance with the fastest convergence and helps diversifying the population to its maximums. The HGA algorithm found a 24-clique after only eight generations.

D. Effect of Migration Rate Variation on Results

In this set of experiments, we fix the mutation rate with the value found greedily in the previous experiment. We then vary the migration rate with the parameters: 1%, 5%, 10%, 20%, 30%, 50% and 90%. Table 3 shows the parameters chosen applied on the same graph (Keller5).

Parameters	Value
Population size	100
Maximum iteration	50
Mutation	1%
Crossover rate	100% (uniform crossover)
Migration rates	1%, 5%, 10%, 20%, 30%,

Table 3 Parameters for testing migration rate

The effect of a high migration rate is the replacement of good solutions obtained by the HGA, whereas a low migration rate injects not enough new seeds in new population. The experiments show that the optimum migration rate is reached at 10% and the maximum clique is found after 28 generations (ground truth maximum clique=27). Figure 5 shows the minimum, the average and maximum fitness score for each chromosome for 10% migration rate.

In this graph, we can observe that the maximum fitness increase by levels, which means that new maximal cliques are found. After one or two generations the stabilization phase is reached and the GA continued graph exploration. The results show that the migration process helped the genetic algorithm to escape from local maxima and find the maximum clique by adding more graph exploration capability.

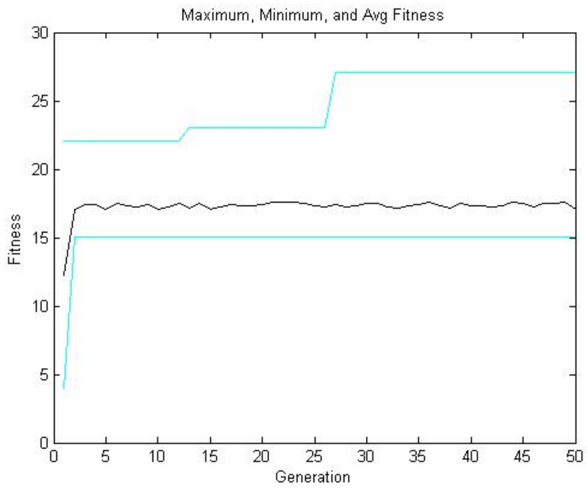

Figure 5 Migration rate 10%

E. Effect of Migration with Variation of Population Size and Number of Generations

In these experiment we test the effect of migration on a variation of the population size and maximum number of generation. We compare the effect of migration on different parameter setup. Table 4 shows the parameters for this set of experiments.

Parameters	Value
(Population size / Maximum iteration / migration)	(10/1000,y), (10/1000,n), (50/200,y), (50/200,n), (100/100, y), (100/100,n), (200/50,y), (200/50,n)
Mutation	1%
Crossover rate	100% (uniform crossover)
Migration rates	10%

Table 4 Parameters for testing effects of migration

The results indicate that for each configuration, applying our migration mechanism increase the size of maximum clique found and also accelerates the convergence toward the best solution.

F. Effect of fitness Sharing on Results

In these experiments we test the effect of fitness sharing on same graph. The parameter σ_{share} from equation (2) represents the number of niches. We vary the number of niches with the values: 3, 7, 10, 20. Table 5 shows the parameters used for this set of experiments.

Parameters	Value
Population size	10
Maximum iteration	1000
Mutation	1%
Crossover rate	100% (uniform crossover)
Number of Niches (σ_{share})	3,7,10, 20

Table 5 parameters for testing fitness sharing

A high number of niches (σ_{share}) allow diversifying the population by reducing the fitness value of individual from the same niche. This way, the chance to promote other individuals from other niches is higher.

We obtained the best results in this series of experiments with the parameter σ_{share}=20. the maximum clique is found after 170 generations. Figure 6 shows the minimum, the average and maximum fitness score for each chromosome for σ_{share}=20.

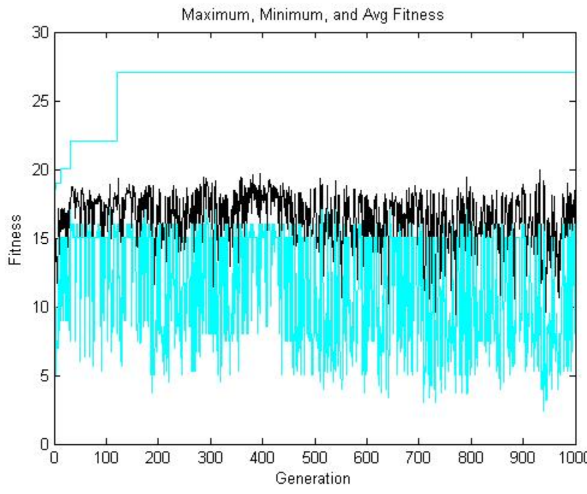

Figure 6 σ_{share} =20

The fitness sharing method also plays a function of graph exploration by reducing the fitness values of individual in the same niche. Then other individual from other niches are promoted in the population. Finally we have shown that fitness sharing method is also well suited for graph exploration and helps to find the maximum clique.

V. CONCLUSION

For the maximum clique problem, we have proposed a solution that implemented a migration mechanism composed of two subpopulations. The main sub-population is specialized on data exploitation and the secondary subpopulation is specialized on data exploration.

Our results have shown that both sharing and migration improve the convergence and results of the genetic algorithm. We have shown that migration proves to be a valuable tool in converging on the maximum, or high maxima relatively quickly. We have also discussed the potential usefulness in using sharing to find multiple maximum solutions in the same problem. We also found that in order to maximize the difference and prevent fast convergence of the minimum fitness value, a 1% mutation rate appears to give the best results. The migration rate was also empirically derived to be best at 10% migration.

These experiments lay groundwork for potential research in the future. We used a non-evolutionary approach by adding a second sub-population, which serves the main population by generating random new seeds. Other improvements can be made by developing a more specific heuristic local search algorithm for the secondary sub-population dedicated to graph exploration.

There are more improvement possibilities by developing a co-evolutionary method where instead of generating new seed at each iteration, the second sub-population evolves at the same time with the main sub-population. Migration is maintained at every generation for faster convergence. Finally a cultural co-evolutionary approach would give feedback from the main sub-population to the secondary sub-population, to help building better candidate for exploration.

REFERENCES

[1] Cheng, James, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. "Finding maximal cliques in massive networks by h*-graph." Proceedings of the 2010 international conference on Management of data, pp. 447-458. ACM, 2010.

[2] Barricelli, Nils Aall (1954). "Esempi numerici di processi di evoluzione". Methodos: 45–68.

[3] Holland J. H. Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, MI, 1975.

[4] Bron, Coen, and Joep Kerbosch. "Algorithm 457: finding all cliques of an undirected graph." Communications of the ACM 16.9 (1973): 575-577.

[5] Tomita, Etsuji, Akira Tanaka, and Haruhisa Takahashi. "The worst-case time complexity for generating all maximal cliques and computational experiments." Theoretical Computer Science 363, no. 1 (2006): 28-42.

[6] Robson, John Michael. "Algorithms for maximum independent sets." Journal of Algorithms 7.3 (1986): 425-440.

[7] Yampolskiy, R., P. Anderson, J. Arney, V. Misic, and T. Clarke. "Printer model integrating genetic algorithm for improvement of halftone patterns." Western New York Image Processing Workshop (WNYIPW). 2004.

[8] Ashby, Leif H., and Roman V. Yampolskiy. "Genetic algorithm and Wisdom of Artificial Crowds algorithm applied to Light up." 16th International Conference on Computer Games (CGAMES), pp. 27-32. IEEE, 2011.

[9] Hindi, Musa M., and Roman V. Yampolskiy. "Genetic Algorithm Applied to the Graph Coloring Problem." Midwest Artificial Intelligence and Cognitive Science Conference, p. 60. 2012.

[10] B. Carter and K. Park. How good are genetic algorithms at finding large cliques: an experimental study. Technical report, Boston University, Computer Science Department, MA, October 1993.

[11] K. Park and B. Carter. On the effectiveness of genetic search in combinatorial optimization. Proceedings of the 10th ACM Symposium on Applied Computing. ACM Press, 1995.

[12] Marchiori, Elena. "A simple heuristic based genetic algorithm for the maximum clique problem." Symposium on Applied Computing: Proceedings of the 1998 ACM symposium on Applied Computing, vol. 27, pp. 366-373. 1998.

[13] E. Marchiori. Genetic, iterated and multistart local search for the maximum clique problem". Applications of Evolutionary Computing. Berlin, Germany: Springer-Verlag, 2002, LNCS 2279, pp. 112–121.

[14] Cantu-Paz, E. (1999) Migration policies and takeovertimes in parallel genetic algorithms. IlliGAL Technical Report No. 99008. University of Illinois at Urbana-Champaign.

[15] Cantu-Paz, E. A survey of parallel genetic algorithms. Technical report 97003. University of Illinois at Urbana-Champaign, May 2007.

[16] Berman, Pioa, and A. Pelc. "Distributed probabilistic fault diagnosis for multiprocessor systems." Fault-Tolerant Computing. FTCS-20. Digest of Papers. 20th International Symposium. IEEE, 1990.

[17] J. MacWilliams and N. J. A. Sloane, The Theory of Error Correcting Codes," North-Holland, Amsterdam, 1979.

[18] Sloane, N. J. A. "Unsolved problems in graph theory arising from the study of codes." Graph Theory Notes of New York 18 (1989): 11-20.

[19] J. C. Lagarias and P. W. Shor, Keller's Cube-Tiling Conjecture is False in High Dimensions," Bulletin AMS, 27(2), pp. 279-283.

[20] M. Brockington and J. Culberson, Camouflaging Independent Sets in Quasi-Random Graphs." Working Paper, Second DIMACS Implementation Challenge, 1993.

[21] T. N. Bui and P. H. Eppley, A Hybrid Genetic Algorithm for the Maximum Clique Problem." Proceedings of the 6th International Conference on Genetic Algorithms (ICGA), Pittsburgh, PA, Morgan Kaufmann, 1995, pp. 478-484.

[22] Goldberg, David E., and Jon Richardson. "Genetic algorithms with sharing for multimodal function optimization." Proceedings of the Second International Conference on Genetic Algorithms and their application, pp. 41-49. 1987.