P-Median: A Performance Analysis

María Beatríz Bernábe Loranca[1], Jorge Ruiz Vanoye[2], Rogelio González Velázquez[1], Marco Antonio Rodríguez Flores[1], Martín Estrada Analco[1]

[1]Computer Science Department, Benemérita Universidad Autónoma de Puebla, México
beatriz.bernabe@gmail.com, marco89_rf@hotmail.com, rogelio.gzzvzz@gmail.com, mestrada@cs.buap.mx
[2]Universidad Autónoma del Carmen
Ciudad del Carmen Campeche, México
jorge@ruizvanoye.com

Abstract. This work approaches the P-median problem with a partitioning around medoids methodology. This problem has been extensively studied because of the multiple applications and its NP-hard nature; therefore several are the efforts to find optimal solutions.
In this paper we consider two case studies: 1) instances from OR-Library and 2) geographical objects from the metropolitan zone of Toluca, Mexico. The methods we use to work with these data are: Partitioning Around Medoids (PAM), Variable Neighborhood Search (VNS), Simulated Annealing (SA), P-Median with MATLAB (PML) and finally Tabu Search (TS).

Keywords: P-median, partitioning, clustering, algorithms.

1. Introduction

In territorial partitioning, two models are the most used: the location-allocation and the set partitioning models. These models seek to group small geographical areas called basic units in a given number of bigger geographical clusters, named territories. The territorial partitioning problem can be modeled like a P-median problem with certain restrictions under the concept of partition, this is, if $\Omega = \{x1, \ldots, xn\}$ is a finite set with n objects we wish to classify and let $k < n$ be the number of classes where we want to group the objects; a partition $P = (C,, \ldots, Ck)$ from Ω in k ses$C1, \ldots, Ck$, is characterized by the following conditions:

$$1. \Omega = \bigcup_{i=1}^{K} c_i$$

$$2. C_i \cap C_j = \emptyset, \qquad for\ every\ i \neq j$$

The P-median problem consists in finding the best configuration of facilities to attend the population's demand in the best way [1]. Given a set of 1000 nodes (coordinates) in the cartesian plane, where every node possesses a certain demand that must be fulfilled, 10 service providers must be installed to satisfy this demand at the minimum cost. The cost can be determined in a proportional way to the distance between nodes.

Given the characteristics of this problem, the formulation to solve it was established in [2]. The problem was named p-median. This problem locates p facilities to minimize the total distance between the demand points and their nearest facilities [1]. For this we solve the allocation problem and minimize the distance and demand of the nodes. The computational complexity of this kind of problems requires using approximate methodologies to give a satisfactory response in regard to quality and time. In this point, two kinds of data have been processed with different heuristic variants based on the P-median model.

2. K-medoids and P-Median

The K-medoids and the P-median problems are similar to each other, while the k-medoids problems deals with cluster analysis to form groups or classes of similar objects, the P-median problem is a facility location problem where the goal is to place facilities on a geographical space to satisfy a certain demand in the area while minimizing transport costs and/or other logistical restrictions (like demand values). When K-medoids methodologies like Partitioning Around Medoids (PAM) are applied to group geographical data, they are basically solving a basic P-median problem where the main goal is to minimize distances between objects (similarity) without other logistical restrictions (uncapacitated). In this paper we employ benchmark matrices from OR-Library for uncapacitated p-median problems and real geographical data from the valley of Toluca, Mexico.

In the section below, we present elementary notions about PAM and the P-median problem and the metaheuristics we have employed in our analysis.

2.1. Partitioning Around Medoids

In the partitioning around medoids methods, PAM has a good reputation because it’s capable to achieve good results. It was developed by Kaufman and Rousseuw [3]. This method assumes that n objects exist and a representative object is determined for every group to find K-clusters (groups), this representative is a medoid. After selecting the K medoids, the algorithm tries to analyze the possible pairs of objects, such that every object is grouped with the most similar medoid. The cost is calculated for each combination according to the defined clustering measurement to determine the best configuration that will be used as the starting point for the next iteration. The complexity of a single iteration is $O(K(n-K)^2)$, therefore the computing time is inefficient for high values of n and K.

PAM Algorithm

```
s ← InitialSolution()
change ← true
While change do
For each medoid m in s do
  For each non-medoid o do
    s' ←Swap(m, o)
    If Cost(s') < Cost(s) then
      s ←s'
      change ← true
    else
      change ← false
    end if
  end for
end for
end while
Return s
```

2.2. P-Median

The territorial and logistical problems have a wide range of applications; from political districting to social services and commercial territories location, etc. In the related literature, geographical compactness and connectivity criteria are employed to measure the quality of the solutions. According to Kalcsics et al. [4], a territory is geographically compact if its shape is round and isn't distorted; however an exact definition doesn't exist. Generally, the sum of the distances between the basic units (objects) and their representative medoid is the compactness measurement. This is how this problem can be modeled like a P-median one [5, 6]. The P-median problem can be expressed as follows:

A finite set of objects must be partitioned in exactly p groups. Each of these groups is characterized by one of its objects that was selected to be the median of the group. A distance is specified for every pair of objects and the sum of the distances between objects and their respective medians must be minimized. Its formulation is the following:

Let $N = \{1, \dots n\}$ be the set of indices for the clients and $J = \{1, \dots m\}$ the set of indices for the potential locations. Typically *N* = *J*. For every *(i,j)*, *i* element of *N*, *j* element of *J*. Let C_{ij} the cost of assigning client *i* to the median located in *j*. The following decision variables are defined:

$$Y_j = \begin{cases} 1 \; If\ the\ median\ is\ located\ in j \in J \\ 0 \qquad\qquad\qquad\qquad otherwise. \end{cases}$$

$$X_{ij} = \begin{cases} 1 \text{ If } the\ client i \in N is\ assigned\ to\ the\ median\ located\ in\ j \in J. \\ 0 \qquad otherwise. \end{cases}$$

Then, the P-median problem is formulated in the following way:

$$Min \sum_{i \in N} \sum_{j \in J} C_{ij} X_{ij} \qquad (1)$$

Subject to:

$$\sum_{j \in J} X_{ij} = 1 \qquad \forall i \in N \qquad (2)$$

$$\sum_{j \in J} Y_{ij} = p \qquad (3)$$

$$X_{ij} \leq Y_j \qquad \forall i \in N, j \in J \qquad (4)$$

$$X_{ij} \in \{0,1\}, Y_j \in \{0,1\} \ \forall\ i \in N, j \in J \qquad (5)$$

Restrictions (2) ensure that each client is assigned to one single median. Restriction (3) guarantees that exactly p locations for the medians are selected. Restrictions (4) establish that the clients are assigned to a median only if this one has been selected. Finally, the set of restrictions (5) specifies that the decision variables are binary.

The classical version of the P-median problem was formulated in 1960 as an extension to the simple facilities location problems proposed, however an algorithm of polynomial order doesn't exist to solve the P-median problem, this is why is considered an NP-hard problem and is still studied.

We employ three approximation methods to solve P-median and K-medoids. The heuristic methods are commonly used to approximate NP problems. We chose three metaheuristics that have been extensively used to solve the P-median problem in particular [7].

2.3. Variable Neighborhood Search

VNS is a metaheuristic technique proposed and described in several works by Mladenovic and Hansen [8, 9, 10]. The basic idea is to combine the application of a local search procedure with a systematic search neighborhood change. The algorithm applies the local search to a solution from the neighborhood of the best solution stored (current solution). If it's impossible to improve this current solution, a bigger neighborhood is considered. When a better solution is obtained the process is restarted. It tries to exploit the idea that the local optimums tend to gather only in certain regions. A more recent tutorial can be found in [11].

The following algorithm shows how the solutions are obtained [12].

VNS Algorithm.

```
Input
  Nk: k=1..kmax, neighborhood structures
```

```
Sa: current solution
Sp: neighbor solution of Sa
Sol: local optimal solution

While Stopping Condition not fulfilled do
  k ← 1
  While k <kmax do
    Sp←GetNeighbor(Sa, Nk)
    Sol ←LocalSearch(Sp)
    If Cost(Sol) < Cost(Sa) then
      Sa← Sol
    else
      k ←k + 1
    end if
  end while
end while
Return Sa
```

2.4. Simulated Annealing

Simulated Annealing is a neighborhood search algorithm with a probability of acceptance criterion based on thermodynamics. It's an optimization method inspired on the metal quenching process used around the 500's B.C. The metal quenching process consists of three phases: a warming phase up to a certain temperature; during the second phase the high temperature is maintained, this allows the molecules to be arranged into minimum energy states, followed by a controlled cooling phase to increase the size of its crystals and reduce its defects. The Metropolis algorithm proposed in 1953 [13] is the pioneer of the simulated annealing methods but Kirkpatrick and Gelatt were the first ones to apply it to optimization problems to find solutions for the travelling salesman problem with several cities [14].

SA Algorithm

```
s ←InitialSolution()
T ← $T_0$
g ← 0
While Stopping Conditions not fulfilled (g, T) do
  s′ ←RandomNeighborFrom(s)
  If Cost(s′) < Cost(s) then
    s ←s′
  else if Random(0,1) <exp((Cost(s)-Cost(s′))/T)then
    s ←s′
  end if
  g ←g + 1
  T ←Update(g, T)
```

```
end while
Return s
```

2.5. Tabu Search

Tabu Search emerged from diverse works published in the late 70's. Despite the fact that the main concepts and strategies behind it already existed, it was in 1989 when the name and methodology were formally established by Fred Glover and Manuel Laguna in the homonymous book Tabu Search [15].

TS is a metaheuristic that guides one or more local heuristics to reach zones of the solution space that regular heuristics usually miss. The local heuristic procedure is an operation used to move between solutions within a defined neighborhood, until reaching a local optimum or fulfilling a stopping criterion. A component from TS that distinguishes it from other metaheuristics is the use of adaptive memory that allows a more flexible search behavior because it stores relevant data about the search process, for example, attributes of the current solution to avoid exploring the same zones of the solution space more than once or to guide the search towards unexplored zones. In this way the motor behind TS is an intelligent and strategic guide capable of making decisions instead of relying on randomness and probability like simulated annealing. This is the biggest contrast between TS and most metaheuristics.

TS Algorithm.

```
Input:
  Number of facilities p
  Number of iterations nit
  Number of iterations for second phase nit2
  Number of worse solutions permitted (perturbation)ip
  Tabu Tenurett

pc ← 0
ic ← 1
S ←InitialSolution()
S* ← S
While ic < nit do
    prev_cost← Cost(S)
    Move(S)
    If Cost(S) > prev_cost then
          pc ← pc+1
    end if
    If Cost(S) < Cost(S*) then
          S* ←S
    end if
    If pc > ip then
          PerturbSolution(S)
```

```
            pc ← 0
    end if
    UpdateTabuLists()
    ic← ic+1
end while
S ← S*
CleanTabuStates()
For i← 0 until nit2 do
    Move(S)
    If Costo(S) ←Costo(S*) then
            S*←S
    end if
UpdateTabuLists()
ic ← ic+1
end if
Return S*
```

The initial solution is generated with the Stingy Drop method, which has given great results to generate initial solutions [7, 16].

Whereas the Move function is a modified faster version of the interchange or swap method proposed by Withtaker that evaluates the profit of interchanging a facility with a candidate facility, in our case we randomly select a cluster and its objects are evaluated as potential facilities and the most profitable is swapped with the medoid [17].

3. Experimentation

The experiments made were tested in a Samsung RV415 laptop with 2GB of RAM and AMD® E-350 CPU at 1.60GHz.

The instances we present bellow were evaluated with VNS, PAM, SA, PML (MATLAB) and TS. The first instances are available in [18] and are widely used in literature to test P-median algorithms. The second instances belong to a real map of the Toluca valley obtained from the National Institute of Statistics, Geography and Informatics of Mexico (INEGI). We present our results in tables 1, 2 and 3 in section 4.

3.1. OR-Library – Uncapacitated P-Median

OR-Library is a collection of test data sets for a variety of Operations Research (OR) problems. OR-Library was originally described in J.E.Beasley [19]. We have taken the 40 available uncapacitated p-median data files that range from 100 to 900 nodes. See Table 1 and 2.

3.2. The Valley of Toluca

This map represents the basic geostatistical areas defined by a census from the year 2000 by the INEGI. It contains 469 objects and our results can be seen in table 3.

4. Results

In this section we cover the results we obtained in several test runs proposed for our data sets.

4.1. OR-Library Instances

In tables 1 and 2, we see the results for the OR-Library instances. We can observe that PAM and TS (table 2) attain the best results in comparison with VNS, SA and PML (Matlab), however PML has a clear advantage in regard to computing time.

Table 1. Uncapacitated P-median problems from OR-Library (Worst methods)

Pmed	OR-Library			VNS		SA	
	Nodes	P	Best	Cost	Time (sec)	Cost	Time (sec)
1	100	5	5819	5819	103	6209	28
2	100	10	4093	4341	180	4646	70
3	100	10	4250	4467	180	4785	47
4	100	20	3034	3380	250	3693	93
5	100	33	1355	1664	360	1820	119
6	200	5	7824	7917	330	8349	49
7	200	10	5631	5952	540	6446	104
8	200	20	4445	5204	1003	5536	174
9	200	40	2734	3385	1860	3626	286
10	200	67	1255	1700	1980	1850	907
11	300	5	7696	7803	720	8346	149
12	300	10	6634	7200	900	7717	298
13	300	30	4374	5126	1860	5475	692
14	300	60	2968	3823	1440	3992	71
15	300	100	1729	2464	240	2558	1721
16	400	5	8162	8423	300	8958	220
17	400	10	6999	7651	540	8197	322
18	400	40	4809	5821	2700	6038	505
19	400	80	2845	3747	2760	3881	2036
20	400	133	1789	2647	2040	2755	1909
21	500	5	9138	9557	240	10231	102
22	500	10	8579	9433	300	9802	170
23	500	50	4619	5645	3600	5941	839
24	500	100	2961	3974	600	4065	1440
25	500	167	1828	2726	720	2852	240
26	600	5	9917	10312	60	10869	14

27	600	10	8307	9065	120	9511	25
28	600	60	4498	5664	480	5799	141
29	600	120	3033	4114	780	4176	70
30	600	200	1989	2960	1320	3058	47
31	700	5	10086	10528	70	11157	93
32	700	10	9297	10383	120	10818	119
33	700	70	4700	6007	720	6166	49
34	700	140	3013	4193	1500	4286	104
35	800	5	10400	11037	120	11698	174
36	800	10	9934	9994	180	11544	250
37	800	80	5057	6460	1620	6715	50
38	900	5	11060	11725	180	12252	10
39	900	10	9423	10570	300	11017	25
40	900	90	5128	6632	2460	6803	40

Table 2. Uncapacitated P-median problems from OR-Library (Best methods)

pmed	OR-Lib	PAM		PML (Matlab)		TS	
	Best	Cost	Time (sec)	Cost	Time (sec)	Cost	Time (sec)
1	5819	5819	0	5891	0.039791	5819	2.556
2	4093	4105	0	4118	0.049311	4093	1.672
3	4250	4250	0	4399	0.056247	4250	1.604
4	3034	3046	1	3088	0.083438	3041	5.703
5	1355	1355	1	1378	0.13098	1394	5.928
6	7824	7824	0	8027	0.075893	7824	49.28
7	5631	5645	1	5646	0.14841	5631	21.744
8	4445	4457	2	4472	0.25138	4451	19.764
9	2734	2753	8	2841	0.49444	2804	31.729
10	1255	1263	14	1295	0.83192	1318	25.288
11	7696	7696	0	7721	0.15035	7696	145.137
12	6634	6634	1	6651	0.28253	6634	63.67
13	4374	4374	20	4467	0.81831	4388	48.169
14	2968	2974	56	3013	1.5988	3091	37.845
15	1729	1738	82	1761	2.6418	1858	48.857
16	8162	8162	1	8232	0.27005	8162	222.629
17	6999	6999	2	7019	0.47927	6999	97.449
18	4809	4811	67	4873	1.8845	4840	25.538
19	2845	2859	296	2899	3.6658	2927	29.422
20	1789	1805	600	1866	6.0295	1882	36.45
21	9138	9138	0	9138	0.38812	9138	164.141
22	8579	8669	4	8670	0-74461	8579	58.606
23	4619	4619	160	4694	3.5808	4664	58.606
24	2961	2965	938	3009	7.1164	3093	127.046
25	1828	1844	1608	1896	11.9582	1937	132.722
26	9917	9917	2	10093	0.56943	9917	389.316
27	8307	8307	9	8364	1.0522	8307	68.365
28	4498	4515	605	4579	6.207	4551	35.594
29	3033	3039	2101	3104	12.3188	3181	66.12

30	1989	2009	2208	2037	20.3498	2119	105.318
31	10086	10086	2	10086	0.74536	10086	479.083
32	9297	9301	8	9331	1.429	9310	109.158
33	4700	4703	1495	4798	9.7708	4735	47.558
34	3013	3026	4685	3097	19.7808	3168	119.309
35	10400	10400	2	10406	0.96892	10400	413.429
36	9934	9934	10	9954	1.8497	9934	141.098
37	5057	5064	2092	5118	14.4414	5278	68.316
38	11060	11060	8	11153	1.2062	11060	86.544
39	9423	9423	13	9451	2.3816	9423	99.102
40	5128	5138	5076	5190	20.838	5214	76.852

4.2. Toluca Valley

In table 3, we present our results obtained for 15 instances using our second data set. We can see that PAM and TS obtain the best results compared with the rest of our algorithms. However because of its intensification, PAM requires a considerable time, for example when P equals 200 PAM finds a solution in 25200 seconds, PML does it in 14.7711 seconds and TS, 34.504.

With this map TS achieves good computing times and the solutions are better than PML, except for instances 10, 11 and 12 and surpasses PAM in tests 2 and 4. After these, VNS is the next method with good results and at last SA. The nomenclature for table 3 is n: test number, P: medians (number of groups), T: computing time in seconds and C: the cost attained.

Table 3. Toluca Valley (460 nodes)

n	P	VNS		PAM		SA		PML (Matlab)		TS	
		C	T	C	T	C	T	Cost	T	C	T
1	5	24.24951	14	23.9643	7	25.0138	0	24.3004	0.37133	23.9643	12.767
2	10	16.8784	21	15.837	17	17.5422	1	16.8496	0.97753	15.5434	8.594
3	15	13.8743	28	12.3637	56	14.74491	1	13.4858	1.1002	12.3767	15.238
4	20	12.0909	46	10.4553	120	12.6997	2	11.2325	1.4222	10.3671	12.949
5	30	9.682202	60	8.0539	300	10.2202	2	8.5286	2.1689	8.0743	11.361
6	33	9.416298	53	7.496998	420	9.552296	2	7.9745	2.4193	7.5694	11.887
7	40	8.322398	61	6.435499	660	8.834101	3	6.8658	2.9317	6.5874	10.973
8	67	5.985501	62	4.360599	2880	6.3118	5	4.6986	5.8809	4.5531	11.652
9	80	5.3999	108	3.711501	2880	5.554698	5	4.0329	8.1257	3.9411	12.432
10	90	4.7458	120	3.3553	3600	5.7672	5	3.2601	9.8468	3.5681	14.222
11	100	4.494802	182	3.0419	3603	4.561401	5	2.7308	14.0718	3.2501	13.954
12	133	3.419801	241	2.3111	11820	3.6658	6	2.4554	13.7786	2.4568	15.705
13	140	3.360899	256	2.1862	12000	3.4056	6	2.3286	9.5345	2.3209	25.646
14	150	3.113098	257	2.0272	18000	3.1526	6	2.1564	10.1451	2.1443	28.511
15	200	2.221599	579	1.4159	25200	2.2481	8	1.5033	14.7711	1.4999	34.504

5. Conclusions

Our analysis of the PAM, VNS, SA, PML and TS algorithms allows us to see the strategies that achieve the best results.

From tables 1 and 2, we clearly see that PAM is the one that obtains the most optimal solutions but its computing time is excessive for instances with several nodes and high values of P. In table 3, for example, we see that PAM takes 25200 seconds to return a solution with 469 nodes and 200 P. In this matter, is clear that PML (MATLAB) works faster by returning a solution in less than 15 seconds for the biggest instance.

On the other hand, TS reaches the best known solutions for the OR-Library instances in most of the cases. PML gives good results for certain instances (for example test run 31 in 0.74536 seconds) but the cost of the solutions is inferior to the costs achieved by TS in most instances and the computing time for TS is a clear improvement from PAM. This can be observed in table 2, however the computing time for TS can be dramatically reduced according to the number of iterations, for example, test 40 finished in 76 seconds while instance 35, which is smaller, in 413 seconds.

VNS and SA present the same issue, their computing times rely on the parameters passed to the algorithms and could respond in much shorter times with the appropriate values, however for these two we have used first improve strategies (the first improving move is selected). This strategies sacrifice optimality for speed, therefore despite our attempt to give them more time to reach a solution (high parameters) they have given the worst results in both tests, this tells us that completely random strategies are unreliable even with extended execution times.

With TS we have implemented best improve strategies (the best move is chosen) while maintaining a good performance, for example for the smaller instances of OR-Library (100 to 300 nodes) the algorithm handles 40000 to 70000 iterations in less than 5 seconds.

PAM relies on a thorough search until no improvement is made having a big influence on the speed to find a solution for big values of P and more than 400 nodes. This makes it the best choice only for small problems to find optimal solutions in competitive times.

In table 3, the map of Toluca, a small sized problem TS is the best choice in terms of time versus optimality because it surpasses the cost of the solutions obtained by PML in several occasions under 35 seconds and in tests 2 and 4 finds better solutions than PAM.

In conclusion the best improve strategies are the most adequate to reach optimal solutions, however a balance should exist between random and intensification strategies to reduce the effect on the performance like we did it in our TS algorithm. Our current challenge is to improve this algorithm to reach the best known solutions for the hardest p-median problems from OR-Library while maintaining a good performance.

References

1. Daskin, M. S.: Network and Discrete Location: Models, Algorithms and Applications¸, John Wiley and Sons, Inc., New York (1995)
2. Hakimi, S.: Optimum location of switching centers and the absolute centers and medians of a graph. Operations Research, 12, 450-459 (1964)
3. Kaufman, L., Rousseeuw, P.: Clustering by Means of Medoids. In: Y. Dodge, Editor, Statistical Data Analysis Based on the L1-Norm and Related Methods, Amsterdam: North-Holland, 405-416 (1987)
4. Kalcsics, J., Nickel, S., Schröder, M.: Towards a Unified Territory Design Approach: Applications, Algorithms and GIS Integration. TOP, 13, 1–74 (2005)
5. Hess, S.W., Samuels, S.A.: Experiences with a sales districting model: criteria and implementation. Management Science, 18, 41– 54 (1971)
6. Zoltners, A., Sinha, P.: Sales territory design: thirty years of modeling and implementation. Marketing Science 24, 313–331 (2005)
7. Mladenovic, N., Brimberg, J., Hansen, P., Moreno, J. A.: The p-median problem: A survey of metaheuristic approaches. European J Operational Research, 179 (2007)
8. Hansen P., Mladenovic, N.: Variable neighborhood search, Les Cahiers du GERAD 96-49 (1996)
9. Mladenovic, N., Hansen, P.: Variable Neighborhood Search. Computers & Operations Research. 24, 1097-1100 (1997)
10. Hansen, P., Mladenovic N., Pérez M. J.: Variable Neighbourhood Search. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 ISSN 1137-3601, 77-92 (2003)
11. Hansen, P., Mladenovic, N.: Variable Neighborhood Search. In Fred Glover & Gary. A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, 145-170 (2003)
12. Pelta, A.D.: Algoritmos heurísticos en bioinformática. PhD dissertation, Universidad de Granada, España (2000)
13. Metropolis, N., Rosenbluth, A., Teller, E.: Equation of state Calculations by Fast Computing Machines, J. Chem. Phys., 21, 6, 1087-1092 (1953)
14. Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P.: Optimization by Simulated Annealing. Science, 220, 671-680 (1983)
15. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell, MA, USA (1997)
16. Al-khedhairi, A.: Simulated Annealing Metaheuristic for Solving P-Median Problem. Int. J. Contemp. Math. Sciences, 3, 1357-1365 (2008)
17. Resende, M., Werneck, R.: A fast swap-based local search procedure for location problems. Annals of Operational Research, 150, 205-230 (2007).
18. OR-library http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html Retrieved on August 25 2014
19. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society, 41, 1069-1072 (1990)