File placement over a network using simulated annealing

Sandip Sen
University of Tulsa

Keywords: File allocation, transaction cost optimization, simulated annealing

Abstract

The redundant allocation of files over a number of sites in a network to minimize the total transmission and access costs for queries and updates is known to be an NP-complete problem even under a static schedule of transactions. As such, a number of heuristic solution techniques have been proposed in literature to address this problem. We propose a stochastic optimization technique, simulated annealing, to find solutions for the file placement problem. Our experiments verify that good solutions to the problem can be found using this algorithm in a reasonable amount of time. This allows for solving larger sized problems than can be done using most other heuristic techniques.

Introduction

We are interested in the problem of distributed resource allocation over a network of computers. In this paper, we deal with a particular instance of that problem, namely, the file allocation problem (FAP) [2, 3, 9]. There are several variants of the problem. In particular, we are concerned with allocating copies of a single file to a subset of processing nodes in the network so that the sum of transmission and access costs for queries and updates are minimized. This problem has been proved to be NP-complete by Eswaran [4]. Several heuristic techniques have been proposed to solve the FAP problem. Most of these techniques use a mathematical programming approach, and are largely limited in the size of the problem for which they can generate approximately optimal solutions [6, 8]. The FAP is a very important real-life optimization problem for which no good solution techniques are available for large problem sizes, and the development of any such technique would be of great service to the distributed computing systems community. Stochastic algorithms such as simulated annealing, genetic algorithms, etc. have been used successfully on a number of difficult optimization problems in the past decade. We have applied a couple of variants of the simulated annealing [5] optimization technique to solve large sized FAP problems (100 node networks).

Application of the simulated annealing technique is not limited to allocating copies of a single file. Given the appropriate cost function to evaluate potential solutions, this techniques can be applied to a number of variants of the FAP problem (including allocating multiple files, problems requiring accessing two or more copies of a file for increased reliability, problems where the cost function includes storage cost, etc.). The only fundamental assumptions that are made are the following:

1. In processing queries, the site of origin of the query decomposes queries involving more than one file so that each of the subqueries involve only one data file. This assumption makes the distributed FAP problem different from the distributed database allocation problem [1], because, in the latter case, subqueries may be processed at a node different from the query origination node and may involve more than one database. This assumption guarnatees that the access and transmission cost for any individual file is minimized in a redundant allocation schema that minimizes the sum total of the access and transmission costs for all the files. As a result, when this assumption holds, techniques used in this paper can be used for multiple file allocation. The assumption also implies the problem complexity of file allocation will depend primarily on the number of sites in the network.

2. We also assume that write updates all and read accesses the closest of the file copies.

We use the term data fragment to refer to either a whole file or a part of it, copies of which are to be redundantly allocated over the network to minimize access and transmission cost. The optimization problem in file allocation involves a tradeoff between response time for queries and update time to maintain consistency across the database. If there are too few copies of a data fragment, queries involving it may take considerable time to be processed. However, the more copies of a single fragment exists in the network, the more time and cost is incurred in maintaining consistency when any data item belonging to that fragment is updated. The size

of the FAP problem, determined both by the number of processing nodes in the network, and the number of data fragments to be allocated, can be quite large. This, together with the fact that no known polynomial time algorithm exists for the problem, make the domain ideal for the application of stochastic optimization techniques like simulated annealing.

The paper is organized as follows: in the Problem Formulation section we define the problem and develop a solution representation to be used by simulated annealing. In the Simulated Annealing section we describe in detail the versions of the simulated algorithms that we have used to address the problem under consideration. In the Evaluation section we present the different criteria that are used to compare the performance of these algorithms on the problem. In the Results section we present the results of our experiments. In the Conclusions section we highlight the findings of our research and identify a useful extension to the current research.

Problem Formulation

We consider the allocation of m fragments redundantly over n sites in the network so as to minimize access and communication costs for q queries and u updates. We assume the existence of a static schedule for queries and updates. The static schedule includes, for each transaction, its frequency of occurrence and the volume of data accessed from each fragment. The total access and communication cost for fragment i is given by

$$\begin{aligned} C_i = & \sum_{j \in Q_i} (\nu_j * (v_{ij} * T_r + (v_{ij} + 1) * T_d * \delta_{ij} + \\ & P_d(NR_{i,O_j}, O_j))) + \\ & \sum_{k \in U_i} (\nu_k * \sum_{r \in R_i} (v_{ij} * T_w + (v_{ij} + 1) * T_d * \delta_{ij} + \\ & P_d(r, O_j))) \end{aligned}$$

where ν_j is the frequency of transaction j, v_{ij} is the volume of data accessed by transaction j from fragment i, T_d is the transmission delay per packet, T_r is the read time per block, T_w is the write time per block, O_j is the site of origin of transaction j, $NR_{i,x}$ is the nearest site to site x which contains a copy of fragment i, $P_d(m,n)$ is the propagation delay from site n to site m, Q_i and U_i are the sets of queries and updates accessing the ith fragment, R_i is the set of sites containing replica of the fragment i, δ_{ij} is 0 if a copy of the ith fragment exists at the origin of transaction j, and is 1 otherwise.

For the stochastic algorithms to be described in this report, we use the following representation for an element in the solution space. For this problem, the solution space is the set of all possible redundant allocation schema of a single data fragment over the n sites. We use an n-bit string (one bit per site) to represent the allocation schema of a fragment. A value of 1 in the ith string position implies that a copy of the fragment exists in site i.

Simulated annealing

Simulated Annealing is a stochastic optimization technique based on an analogy from statistical mechanics, where a substance is reduced to its lowest energy configuration (or the *ground state*) by a sequence of steps that involve alternate heating and cooling. Cooling leads to low energy configurations, while heating prevents the substance from getting stuck at local minima by raising its energy. As the substance approaches its ground state, it is subjected to smaller changes in temperature, so that a 'good' solution is not easily disturbed. The ground state is reached when the substance settles into a stable state at a very low temperature.

Solving an optimization problem using simulated annealing involves the following four steps:

1. Encoding the points in the solution space (analogous to the *configuration*).
2. Formulating an evaluation function to determine the goodness of the current solution (analogous to determining the *energy value* of the configuration).
3. Deciding a set of moves used to generate a new solution from the current solution (analogous to looking for configurations with lower energy and switching to them in a probabilistic manner).
4. Deciding an annealing *schedule*, i.e., choice of starting temperature, rule for temperature decrements, number of elements examined at every temperature (chain length), and the stopping criteria used to halt the algorithm.

The representation used in our experiments has been described in the previous section. The evaluation function is the cost function developed in Problem formulation section that computes the total access and transmission costs for a given allocation schema. The optimal allocation schema is the one for which the cost function is minimized. Three types of moves are defined in our annealing scheme: the first places a replica of the fragment at a new site, the second removes a replica from a site, and the third moves a replica from an old site to a new one (the *swap* operator). The first two of these moves can be represented by a single move operator called *mutation*, that flips a randomly picked bit. In all our experiments we used either mutation or a swap operator equiprobably to generate a new configuration from the present configuration.

In this paper, we present results of our experiments with two different annealing schedules:a very simple hand-coded scheme, and an adaptive, computationally more extensive scheme. We describe each of these in more detail below.

Fixed schedule

The first of the annealing schedules involves a fixed initial and final temperature setting and a constant factor decrease of temperature. The schedule involves thirteen different temperature levels (determined by a decay factor of 0.7, starting from a maximum temperature of 100 and lowering it to 1). Moreover, at each temperature level, the number of points examined is dependent on the number of sites in the problem, namely n. In the following, v_i gives the cost function evaluation of the allocation schema corresponding to the structure $x(i)$. The algorithm is as follows:

1. *(Start)* Set $T = T_{max}$. Select a point x_c at random and evaluate it.
2. *(Stochastic hillclimb)* Pick an adjacent point x_a at random and evaluate it. Select the new point (i.e., $x_c = x_a$) with probability $\frac{1}{1+e^{-(\frac{v_a - v_c}{T})}}$. Repeat the step k times.
3. *(Anneal/Convergence test)* Set $T = rT$. If $T \geq T_{min}$, go to step 2, otherwise done.

Adaptive schedule

The second schedule we experimented with is an adaptive schedule that involves dynamic determination of temperatures, decrements, and step numbers based on estimating the mean and standard deviation of the function space by finite sampling. The following algorithm is gleaned from [7] which itself summarizes the findings of a lot of other researchers. In the subsequent description of the algorithm we use the following notations:

- σ_∞, σ_t are the actual standard deviation of the function values, and the standard deviation of the function values observed when the schedule is at temperature t.
- E_∞, E_t are the actual mean of the function values, and the mean of the function values observed when the schedule is at temperature t.
- H_∞ is the size of the configuration (solution) space (also called *global accessability*), H_0 is the number of function optima (assumed to be 1 in this paper), H_t is the global accessability at temperature t (a measure of the size of the likely state space).
- h_t is the local accessability at temperature t (a measure of the access to neighboring states from the current state). β is the selection probability of a neighboring state. a_t is the fraction of moves at temperature t that changed the current state. U_t is the average of the positive moves that was accepted at temperature t.
- T_i is the temperature at the ith step of the annealing schedule, t_{stop} is the current estimate of the stopping temperature, and t_{linear} is the current estimate of the highest temperature at which the $\sigma_t - t$ curve becomes linear.

The algorithm is as follows:

1. Estimate σ_∞, E_∞ by sampling r points (we have used $r = 384$ randomly from the solution space).
2. *(Start)* Set $T_0 = m\sigma_\infty, m > 1$ (we chose $m = \sqrt{10}$). Select a point x_c at random and evaluate it.
3. *(Stochastic hillclimb)* Pick an adjacent point x_a at random and evaluate it. Select the new point (i.e., $x_c = x_a$) with probability $e^{-(\frac{v_a - v_c}{T_i})}$. Repeat the step $k_{i+1} = c * \frac{H_i}{h_i}$ times (we have used $c = 4$), where
$$H_i = H_\infty + \int_\infty t\frac{\mathrm{dE}}{t},$$
$$h_i = a_i \ln(\frac{1}{\beta}) + \frac{U_i}{i} + (a_i - 1)\ln(1 - a_i).$$
If $k_{i+1} >$ max_moves, then k_{i+1} =max_moves (we have used a value of 500).
4. *(Anneal/Convergence test)* Set $T_{i+1} = T_i - \gamma\frac{i^2}{\sigma_i}$ if $T_i > \gamma\frac{i^2}{\sigma_i}$. Otherwise, set $T_{i+1} = \frac{T_i}{2}$. Also calculate, $t_{linear} = \sigma_\infty\frac{i}{\sigma_i}$, and
$$t_{stop} = t_{linear}\exp\left(\frac{1}{2} - \frac{t_{linear}^2(H_\infty - H_0)}{\sigma_\infty^2}\right).$$[1]
If $\sigma_i > \theta * i * (E_\infty - E_i)$ or $T_{i+1} > t_{stop}$, go to step 3, otherwise done. We have used $\theta = 0.01$.

Evaluation of the algorithms

One objective of this research was to compare the performance of the above-mentioned variants of the simulated annealing algorithms on FAP instances. In order to perform an objective evaluation, we decided to build a parameterized test case generator to provide us with test cases with desired characteristics. We used the following parameters to control the test cases generated:

- number of clusters, number of sites per cluster
- minimum (25km), maximum (75km) inter-cluster distance; minimum(400km), maximum(600km) intra-cluster distance
- number of queries, number of updates
- minimum (20), maximum (30) frequency of queries; minimum (1), maximum (10) frequency of updates

[1] Double exponential smoothing is used to calculate a reliable estimate of σ_i (to reduce the effects of widely scattered σ values at high temperatures).

- minimum (5) , maximum (10) size of queries; minimum (1), maximum (5) size of updates
- network bandwidth (we have used 1.544Mbps), network latency (we have used 200,000 km/sec).

We assume that nodes are distributed over the network in clusters. Variable values are generated from an uniform distribution over corresponding domains. The size of a problem is decided by the number of sites in the network. For each problem size (we have used 20, 64, and 100 sites), we generated 10 test cases. Number of clusters and number of sites per cluster for the different problem sizes were 5, 8, 10 and 4, 8, 10 respectively. Number of queries and number of updates for the different problem sizes were 35, 75, 200 and 3, 6, 10 respectively. Performance of an algorithm on a test case was obtained by averaging over 10 runs with different random seeds. Performance of an algorithm on a particular problem size was obtained by averaging its performance on all the 10 test cases of this problem size.
We evaluated performance of an algorithm by two criteria: the goodness of the solution that it finds, and how quickly it finds it. To verify the goodness of the solutions obtained, we implemented an exhaustive search algorithm to give us the optimal solution for reasonable size problems (we could run exhaustive search for up to 20 site problems). The second criteria for evaluation measures the average number of solutions evaluated before the algorithm finds the best solution it generates.

Results

Quality of solutions: From our experiments with problems of size 20, we found that both versions of the simulated annealing found the optimal solution to all the test cases in every run. When we experimented with problems of size 64, we found that the best solutions found in the different runs involving the same test case differed for the fixed schedule algorithm. For some problems in almost half the runs, the fixed schedule algorithm failed to reach the best solution it had found in other runs. This indicates that the algorithm was getting stuck at local optimas with increasing frequency. Presumably this deficiency can be overcome by either choosing smaller temperature decrements or by choosing more iterations at each time step (the chain length used were 20, 75 and 125 for problems of size 20, 64 and 100). But the adaptive schedule algorithm automatically chooses a good schedule without this trial and error process which can be inefficient and error-prone. The overall best solution found by both the schedules were identical for every test case. This increases the likelihood that those were indeed the optimal solutions to the respective test cases [2].

[2] For problems of this size, we do not know the optimal solution since it involves exhaustive enumeration.

The adaptive schedule algorithm found the best solution to each individual problem every singly time it was run. This is ample proof that this algorithm can be used very effectively to solve file allocation problems over relatively large networks. The running time of the algorithm for the 100 node problem was about 5 minutes on an IBM RS6000. This suggests that even larger sized problems can be successfully solved using this method. We can even envisage dynamically reconfiguring the network in cases where static schedules are not available, and update and query frequencies and sizes are estimated incrementally at run time.

Speed of Convergence: Figure 1(a) shows the average number of trials (solution evaluations) performed by adaptive and fixed schedules to find the best solution for different problem sizes. The graph shows that the fixed schedule takes approximately one-tenth the number of trials to produce solutions of the same quality as that obtained with the adaptive schedule for problems of size 20. For problems of larger sizes however, this difference is reduced, as the fixed schedule has to be run for increasing number of trials to match the performance of the adaptive schedule in terms of the quality of the solution that it generates. Given a problem of arbitrary size, we can extrapolate the curve for the fixed schedule algorithm in the graph to estimate the average number of trials it would take for the fixed schedule to find the best solution. This number can be used to calculate the chain length for the fixed schedule, thus giving us a complete fixed schedule for that problem size. The schedule for the adaptive method for any problem of any size would be, of course, automatically generated by sampling the function space. For the problem sizes that we have tried, both algorithms exhibit linear increase (albeit with different slopes) in the number of solutions evaluated to get to the best solution. This is in contrast to the exponential increase in complexity of the only algorithm guaranteed to find the optimal solution, namely exhaustive search.
Figure 1(b) shows the progress of the fixed schedule SA towards the best solution over the course of a run. Since the initial structure is chosen randomly, the quality of the initial solution is bad, but the figure also demonstrates the rapid convergence of the fixed schedule SA to the best solution. The performance curve of the adaptive schedule SA is similar in nature.

Conclusions

We used a domain independent stochastic optimization technique to address an important problem in distributed computer systems, namely the problem of allocating data fragments redundantly over a number of sites on a network. We assumed the existence of a static query and update schedule. The results demon-

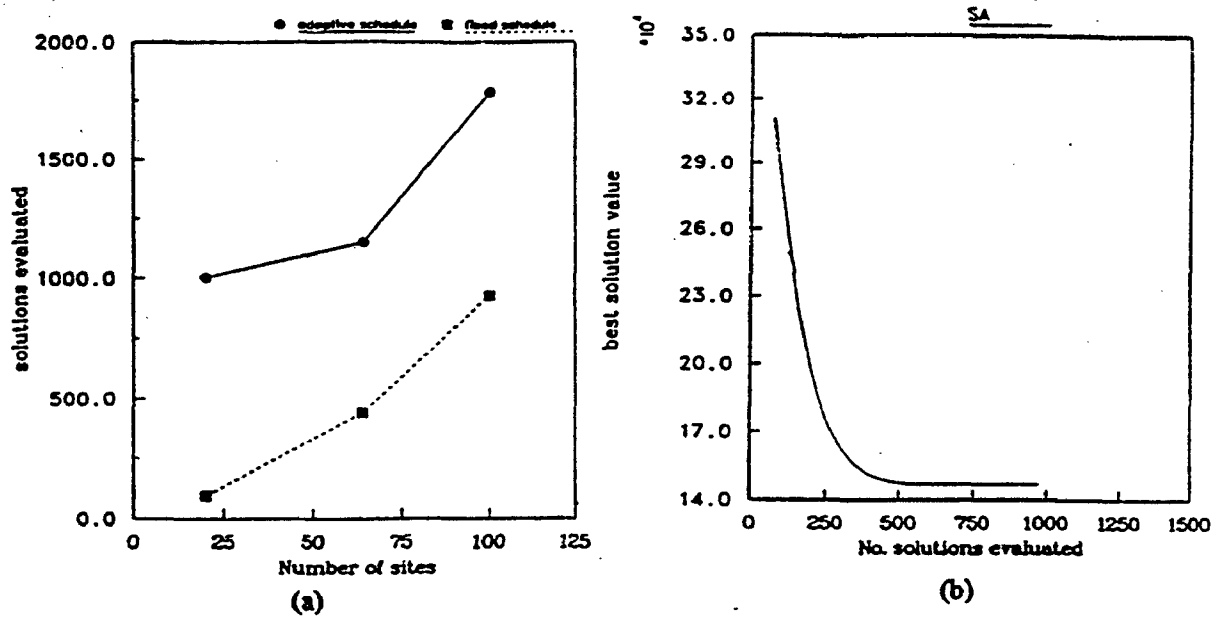

Figure 1: (a) Average number of trials required by fixed and adaptive schedule SA to find the best solution for different problem sizes, (b) Typical run of a fixed schedule simulated annealing algorithm on a problem of size 64.

strate the feasibility of two of these techniques, the fixed schedule and adaptive schedule simulated annealing methods, in finding good solutions to the problem with limited search. A comparison of these two algorithms shows that the adaptive schedule simulated annealing performs more search than the fixed schedule version to find good solutions to problems, but finds them with greater certainty. The effectiveness of these methods is demonstrated by their ability to consistently find good solutions for medium to large sized problems. While most of the previous heuristic methods used in solving the file allocation problems present results with 10-25 node networks [6, 8], we have demonstrated that our approach can easily solve problems with 100 nodes in the network. This fact, together with the likely success of using simulated annealing for even larger sized problems, makes these optimization techniques a viable tool for solving file allocation and related resource allocation problems in distributed computer systems.

Our approach of optimizing the allocation of each fragment independently is based on the assumption of no communication between sites holding required fragments during a transaction. Research in distributed transaction processing suggest that optimal transaction costs can require such communication. We believe that the no communication assumption does not limit the applicability of stochastic methods to the general distributed data allocation problem [1]. We intend to relax this assumption in our future work.

Acknowledgements

I thank Santanu Paul for providing valuable suggestions and insights that contributed to the development of this work.

Author affiliation

Dept. of Mathematical & Computer Sciences, University of Tulsa, 600 South College Avenue, Tulsa, OK 74133; e-mail: sandip@kolkata.mcs.utulsa.edu; phone: 918-630-2985.

References

[1] P.M.G. Apers, Data Allocation in Distributed Database Systems. *ACM Transactions on Database Systems*, vol. 13, no. 3, pp. 263-304, September 1988.

[2] W.W. Chu. "Optimal file allocation in a multiple computer system," *IEEE Trans. Comput.*, vo. C-18, pp. 885-889, 1969.

[3] L. Dowdy and D. Foster, "Comparative models of the file assignment problem," *ACM Computing Surveys*, vol. 14, pp. 287-314, June 1982.

[4] K. Eswaran, "Placement of records of a file and file allocation in a computer network," *IFIP Conf. Proc.*, pp. 304-307, 1974.

[5] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science, 220*(4598), pp. 671-680, 1983.

[6] J.F. Kurose and R. Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems," *IEEE Trans. Computers*, vol. 38, no. 5, pp. 705-717, May 1989.

[7] R.H.J.M. Otten and L.P.P.P. van Ginneken, *The Annealing Algorithm*. Boston, Massachusetts: Kluwer Academic, 1989.

[8] S. Ram and R.E. Marsten, "A Model for Database Allocation Incorporating a Concurrency Control Mecahanism," *IEEE Trans. Knowledge and Data Engg.*, vo. 3, no. 3, pp. 389-395, September 1991.

[9] B. Wah, "File placement in distributed computer systems," *IEEE Computer*, vol. 17, pp. 23-33, Jan. 1984.