ELSEVIER

European Journal of Operational Research 88 (1996) 139–151

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

Mechanisms for local search

E.J. Anderson

The Judge Institute of Management Studies, University of Cambridge, Cambridge CB2 1RX, UK

Received September 1992

Abstract

Local search methods for combinatorial optimization make a series of steps, at each stage improving the current solution by moving to a neighbouring solution. This is usually done by considering the neighbouring solutions one at a time and moving to the first one which gives an improvement (a first-improving method). In this paper we consider whether there are circumstances in which some other strategy will have better performance. In exploring this question we begin by giving a theoretical treatment of a simple model with random objective values at each solution point. We carry out some experiments on the Travelling Salesman Problem and the Quadratic Assignment Problem using varying values of a spread parameter, k. The value of k corresponds to the number of improving solutions looked at before making a move. We also make some conjectures relating the overall performance of the local search method to the average number of solutions which are evaluated before a local minimum is reached.

Keywords: Local search; Optimisation; Heuristics; Travelling Salesman Problem; Quadratic Assignment Problem

1. Introduction

Local search techniques have been very popular as a means of finding reasonable solutions to hard combinatorial optimization problems (see for example the chapter on local search in Papadimitriou and Steiglitz, 1982). Recent interest in this area has also been stimulated by the increasing use of methods such as Simulated Annealing and Tabu Search, both of which are related to straightforward local search. There are innumerable examples of combinatorial problems for which local search methods have been suggested as appropriate, or for which a local search method can be used to improve a solution which has been generated by some other heuristic method.

Local search methods work on the assumption that there is some neighbourhood structure defined on the space of all possible solutions. We shall write X for the set of all possible solutions. There is also a given objective function (or cost), $c(x)$ for $x \in X$. It is desired to find an optimal solution, i.e. an $x^* \in X$ with $c(x^*) \leqslant c(x)$ for all $x \in X$. For each $x \in X$ we define a set of neighbours $N(x)$, a subset of X. We shall assume that the neighbourhood relation is symmetric, so that if x is in $N(y)$ then y is also in $N(x)$. The neighbourhoods define what is 'local' to a given solution point. We will attempt to find an optimal solution by making a series of local moves, and we therefore ought to ensure that we can move throughout X using a sequence of such moves. This property is conveniently described by defining the *neighbourhood graph* which consists of a node for each solution point with an undirected

SSDI 0377-2217(94)00164-8

arc joining two nodes if and only if they are neighbours. Then the condition that we can move in a sequence of steps to any solution point translates into the condition that the neighbourhood graph is connected.

The simplest local search technique is to move from each point to a neighbouring point with a smaller value of the objective function. This process repeats until a solution is reached which is better than any of its neighbours. This can be thought of as a 'local' optimum, and with luck it will not be much worse than the global optimum. An obvious way to improve on this solution method is to run the procedure a number of times starting from different, perhaps randomly generated, initial solutions. In this way we are likely to generate a number of local optima and we can then choose the best of these as our final solution. In this paper we will assume this model of local search and concentrate on the method that is used to select a neighbour with an improved objective function value. Yannakakis (1990) refers to this as the *pivot rule*.

There are two basic methods which have been commonly used to select an improving neighbour:

(a) *Best-improving:* We can look at all the neighbours of a given solution and then move to the best. This is in effect a 'steepest descent' method – we make the local move which produces the greatest improvement. Tovey (1985) uses the term *optimal adjacency* algorithm for this method.

(b) *First-improving:* We can look at the neighbours one by one, taking them in a random order, and move to the first one that we find which is an improvement on the current solution. In effect we are taking a random sample of the neighbours, taking the sample one at a time and without replacement in order to avoid looking at the same neighbour more than once.

There is also another version of the first-improving method which has been found to be effective in practice, which we will call *ordered first-improving* local search (Papadimitriou and Steiglitz use the term *circular* searching). It often happens that all the neighbourhood sets are the same size and that there is a natural correspondence between any two neighbourhood sets. One example of this occurs when solutions to a problem can be represented by binary strings of a certain length, n say. In this case a natural neighbourhood structure can be obtained by defining two solutions as neighbours if they differ in exactly one bit of their binary string representations. Then we can index the neighbours of each point by letting $N_i(x)$ be the string whose i-th component differs from x, $i = 1, 2, \ldots, n$.

For any problem in which neighbours can be indexed in a consistent way, we can define ordered first-improving local search as first-improving local search in which neighbours are considered in their natural order and the sequence is continued from where it left off after a move is made. Thus for example if, in looking at neighbours of x, the first one found which is better is $y = N_{12}(x)$, say, then the first neighbour of y to be looked at is $N_{13}(y)$, and then $N_{14}(y)$, and this is continued, if no improvement is found, until $N_n(y)$ is reached at which point we start again from $N_1(y)$ going on till $N_{12}(y)$, when we can deduce that we have found a local optimum.

The main aim of this paper is to explore the difference between first-improving and best-improving local search, but the paper may be of interest for other reasons. First our analysis and experimental results suggest that the average number of different solutions looked at in a single run of a local search method may be an effective predictor of the performance of local search – if this conjecture is correct it will be helpful in deciding between two alternative neighbourhood structures for a given problem.

It is not straightforward to obtain conclusive results from experiments on local search methods for combinatorial optimization: there are often a bewildering variety of factors which may have an influence, and there are real difficulties in assessing how the choice of best method may be affected by the amount of computing time available, and how much preliminary 'tuning' of parameters is proper. In this paper we are considering a very straightforward issue and this will make it easier to address some of these difficulties. The approach we have used in our computational work may be of interest to others working in this field, as indicating one possible way to deal with

the difficulties of presentation and analysis of results.

In this paper we begin by discussing some idealised models of the local search method to guide us in understanding the different types of behaviour that are observed in using different local search methods. We consider both a random model and also a model with structure. Then we report the results of an experimental study of different local search methods for the Travelling Salesman Problem and the Quadratic Assignment Problem. Finally we analyse the expected number of solutions evaluated before a local minimum is reached and explore how this varies with problem size.

2. Random values

In this section we consider a problem in which the values of each solution are assigned randomly, being drawn from some probability distribution. Then it becomes possible, at least in principle, to carry out some analysis to find the expected value of the local optimum which is reached starting from a given point and knowing the structure of the neighbourhood graph. In these circumstances, for reasons which we will discuss below, we would expect best-improving local search to do better than first-improving local search.

We begin with a result in which we assume that X is an infinite set of solutions. In doing this we depart from the usual assumptions for a combinatorial optimization problem, but nevertheless the result is of interest.

Theorem 1. *Suppose that each solution has the same number, N, of neighbours, the neighbourhood graph is an infinite tree, and that the values assigned to solutions are independent identically distributed random variables. For a given starting point, x, let $V_B(x)$ be the expected value of the (locally optimal) solution reached using best-improving local search, and let $V_F(x)$ be the expected value of the solution reached using first-improving local search. Then $V_B(x) < V_F(x)$.*

Proof. We will prove this result, not by directly comparing the values of the best solution found under the two strategies, but by looking at the number of different solutions that are evaluated. Since the solution values are i.i.d. random variables the final result, which is the best solution evaluated, is a random variable with a distribution which only depends on the total number of solutions evaluated. We will show that the number of solutions evaluated is greater under best-improving local search.

First consider first-improving local search. It is easy to see that the local search will stop if evaluations 2 up to $N+1$ are all worse than the first evaluation, or if at any stage after this a sequence of $N-1$ successive evaluations are performed without finding an improving solution. In these circumstances we will have completely evaluated a neighbourhood of the current best solution.

Now consider best-improving local search. In this case we do not immediately move to the best solution we have so far found, but only after a round of $N-1$ evaluations are made (or N evaluations at the start). The consequence is that we may make a sequence of $N-1$ evaluations without making an improvement and still not be forced to stop. For example consider the situation shown in Fig. 1 in which $N=3$ and the local search begins at node 1 (the node numbers are shown inside the circles). Suppose that the numbers shown beside each node are a particular realisation of the function values. If nodes are looked at in node order then there will be successive evaluation of nodes 6 and 7 which both have function values worse than the 'current' best solution which is node 5. This cannot happen with first-improving local search in which evaluation of two successive solutions worse than the current best solution causes the search to stop. Notice also that best-improving local search cannot be forced to stop after less than $N-1$ unsuccessful evaluations.

In this way of looking at the problem both procedures amount to the successive evaluation of i.i.d. random variables until some stopping rule operates. We have shown that the stopping rule for best-improving local search only stops when

that for first-improving also stops, and moreover there are situations in which first-improving stops but best-improving does not. Thus the number of evaluations made under best-improving local search is (stochastically) greater than that under first-improving local search and the theorem is established. □

The model we have used here, which is an infinite tree, will be appropriate for a large neighbourhood graph in which there are no short cycles (paths through the graph which return to their starting point). Since every step is an improving one the probability of a large number of steps being taken will be small, and thus the structure locally is all that is important. Provided that there are no short cycles, the existence of long cycles is unlikely to influence the behaviour of the two algorithms and we will be able to conclude that the best-improving algorithm will produce a better expected final value. Nevertheless this theorem is rather weaker than we would like. We conjecture that the result remains true for any problem in which each solution has the same number of neighbours and the neighbourhood graph makes no distinction between different nodes (so in this sense all solutions are equivalent).

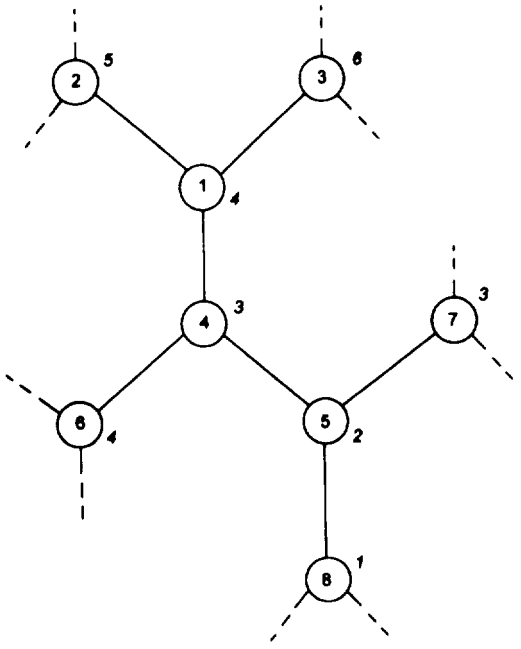

Fig. 1. An example with $N = 3$.

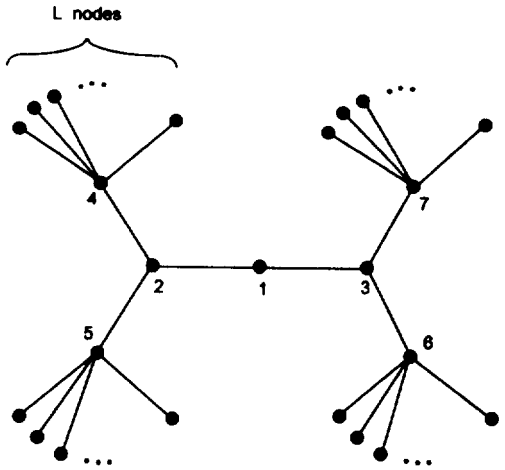

Fig. 2. The neighbourhood graph for Example 1.

In other words we do not believe that the tree structure restriction is necessary for the theorem to hold. On the other hand the restriction that each solution has the same number of neighbours is a necessary condition, as the following example makes clear.

Example 1. Consider a neighbourhood graph having the form of Fig. 2 in which local search always starts from node 1 and in which node values are set as one of 3, 2, 1 and 0 with probabilities of $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{2} - \varepsilon$ and ε respectively. We suppose that the values of L and ε are chosen (large and small respectively) so that the probability of a value 0 occurring at one of the nodes 1, 2, 3, 4, 5, 6, or 7 is small enough to be ignored but there is almost always a 0 value amongst each of the 4 sets of L neighbours of the nodes 4, 5, 6 and 7.

If node 1 has a value 0 then both algorithms stay at node 1. If node 1 has value 1, then both algorithms will move to node 2 or 3 if they contain a 0 but otherwise stay at node 1. In either case the eventual value will be the same for both algorithms. If node 1 has value 2 then there is the possibility of the two algorithms ending with different values, but this can only happen if one of nodes 2 and 3 has the value 1 and the other has

the value 0. Since ε is very small, the probability of this occuring is very small.

If node 1 has value 3 then there are more possible outcomes, including the possibility that first-improving does better than best-improving. There are a number of cases in which a zero value occurs amongst nodes 2 to 7, but these happen only with a very small probability. The significant case occurs when one of nodes 2 and 3 has value 2 (say node 2) and the other has value 1. The best-improving algorithm will then move to node 3 and end with a value of 1 (unless one of nodes 6 and 7 has a value of 0 which happens with a very small probability). With probability $\frac{1}{2}$ the first-improving algorithm will do the same and with probability $\frac{1}{2}$ it will move to node 2. In this case it stops at node 2 if neither of nodes 3 and 4 have a value of 1 or 0. On the other hand if it does move to one of nodes 3 and 4 then because of the large size of k there is a very high probability that it will end with a value of 0. Thus, if there is a difference between the two policies, then there is a probability of approximately $\frac{1}{4}$ that it favours best-improving and a probability of approximately $\frac{3}{4}$ that it favours first-improving. Consequently the first-improving algorithm will do better overall.

We have carried out some numerical tests to verify this reasoning. Using a simulation we can show that, for this example, with $\varepsilon = \frac{1}{32}$ and $L = 60$ the first-improving algorithm achieves an expected value of about 0.92, while the best-improving algorithm achieves an expected value of about 0.93.

It is important to realise that the result of Theorem 1 applies to a situation when just one run of a local search procedure is used. In these circumstances, under the assumption of random solution values, a better result will be obtained by using best-improving local search. But as we pointed out in the introduction, it would be normal to take a number of runs from different random starting positions. In this case the apparent advantage of the best improving method will disappear. Since there is no structure within the problem, the quality of the final solution will depend only on the total number of evaluations carried out. So if some finite amount of computer time is available, there will be no difference between the two methods – since there are fewer evaluations required for a single run of the first improving method there will be time for more runs in total. In Section 4, below, we will return to the issue of how to compare two different methods in circumstances where there is only a finite amount of computer time available.

3. Local search for problems with structure

It is clear that the model we have considered, in which the values for each solution point are i.i.d. random variables, leaves out those aspects of the approach which make it successful in practice. If it were really the case that there was no relation between the value of neighbouring solutions, then one might as well proceed by taking an entirely random sample from all the possible solutions, evaluating them and then choosing the best. There are two reasons for using local search in practice. One reason is that it is often easier to calculate a *change* in solution value obtained when moving to a neighbour, than it would be to calculate the solution value starting from scratch. Consequently local search offers a potential time saving over a random sampling method. The second and more important reason is that local search can exploit the fact that neighbouring solutions have similar values.

It is often misleading to think of neighbourhood graphs as having a planar structure, but a model with these properties does make it clear how local search can be effective when the solution values are well-structured. Consider a model in which the solutions can be described by their position (x, y), with x and y integers, and where we take (x_1, y_1) and (x_2, y_2) as neighbours if

$$|x_1 - x_2| + |y_1 - y_2| = 1.$$

Suppose that the value of solution (x, y) is $|x| + |y|$. We call this the *inverted pyramid model*. Then each step of a local search procedure will reduce one of $|x|$ and $|y|$ and the procedure will rapidly reach the optimum $x = y = 0$.

Now consider the behaviour of the first-improving and best-improving methods for this model.

Once underway the best-improving method will consider three neighbours at each step, whereas first-improving will consider at most two neighbours (except when one of x and y is 0). Thus the first-improving method will reach the optimum more quickly.

As another example of a structured problem consider a pattern matching problem in which the solution is defined by a string $(x_1, x_2, \ldots, x_n)$ of 0's and 1's and the objective function is $\Sigma(x_i - a_i)^2$, where the values $(a_1, a_2, \ldots, a_n)$ constitute the pattern which we wish to match. We take two solutions as neighbours if they differ at only one position in the string. Here again the first-improving method will reach the optimum much faster than the best-improving method. The reason for this is that if there is an improving move, say changing the string element at position m, then the same improving move is still available after some other move has been made. There is thus no point in expending computational time looking at a number of possible improvements to find the best: this 'best' move will still be available to us later on in the search procedure even if we do not take it now. Something similar is also happening in the inverted pyramid model.

This gives one reason why we might expect the first-improving method to be superior in real problems – it may move more quickly to a local optimum. A second reason is that it may be less likely to become stuck in a relatively poor local optimum. To understand this type of behaviour consider a model in which there is a mixture between an inverted pyramid structure and a random element. Suppose for example that the solution value at (x, y) is $|x| + |y| + \varepsilon_{xy}$ where ε_{xy} is a random variable taking the value -3 with a small probability, and otherwise being 0. Thus we have an inverted pyramid of $|x| + |y|$ with the occasional dent in it. In Fig. 3 we illustrate this behaviour at the point (10, 10) assuming that $\varepsilon_{10,10}$ is equal to -3.

It is not hard to see that in these circumstances the best-improving method leads to the solution becoming stuck in one of the false local minima more often than the use of the first-improving method. For example, using the data of the figure, starting at the point (11, 10) we are

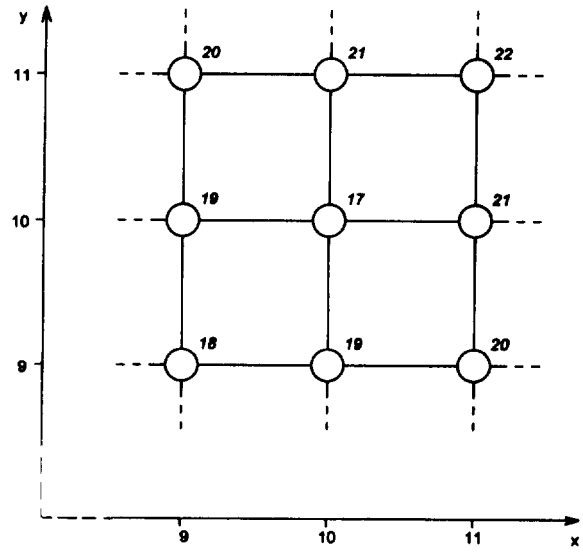

Fig. 3. Solution values on a grid.

bound to move to (10, 10) using the best-improving method, but we have a probability of only $\frac{7}{12}$ of doing so using the first-improving method from this starting point. The consequence is that in this case the use of the first-improving method will lead to a better final solution value than the use of the best-improving method.

It may be interesting to note that in this situation even better performance is obtained by taking a *worst-improving* method. This is defined as a method which selects the worst solution from the set of neighbours of the current solution which improve on the current solution. This is a kind of 'shallowest descent' policy which can be expected to take a long time to reach a good solution, but minimizes the chance of falling into a local optimum too early.

4. The experiments

There are obviously many pivot rules which are intermediate between best-improving and first-improving. It is convenient to define a single spread parameter k (which is a positive integer) in order to parameterise a whole class of improving policies. We make a single improving step by considering neighbours of the current solution

one at a time without replacement until we have found k improving neighbours (or until we have considered all the neighbours of the current solution), and then we move to the best of these. Using this definition, a value of k equal to 1 gives rise to the first-improving method.

Our experiments have been carried out on the Travelling Salesman Problem (TSP) and on the Quadratic Asignment Problem (QAP). These are both intensively studied combinatorial optimization problems, which makes them a natural choice for our purposes. For the TSP we consider a 100-city problem, with a Euclidean distance matrix. For the QAP we consider a 20 location problem. We have also carried out experiments, which we do not include here, on a number of 50-city TSPs and also some 30 location QAPs. We would have preferred to use larger problems, but if we had done so it would not have been possible to carry out such comprehensive experiments.

We begin by testing different values of the spread parameter k using a 2-Opt edge exchange neighbourhood. This is the simplest neighbourhood which has been found to be effective for the symmetric TSP. We generated a problem with 100 city locations drawn at random from a square. Starting solutions were generated at random. This is the approach that has been usually followed when using multiple local search, though Wong and Morris (1989) argue for a more systematic selection of starting points. In Fig. 4 scatter plots are given for 50 runs with spread parameter $k = 1$ (first improving), and for 50 runs with $k = 16$. These show the length of the final tour reached and the number of different tours which were evaluated along the way. This latter quantity is the dominant term in the total computation requirement and we have chosen to use it as a surrogate for computation time, since this will make it easier to compare our results with those of other researchers.

As can be seen from the scatter plots using $k = 1$ is much faster, but the use of a larger spread parameter leads, on average, to slightly better solutions. Nevertheless, for any fixed amount of computing time we could make a far greater number of runs with $k = 1$ than with $k = 16$ and this might well lead to a better solution being found. In this sense the data we have so far looked at does not make it clear which is the better value for k.

In order to address this question we suppose that there is only time for a fixed number of evaluations, and ask what is the expectation of the best value achieved over a succession of runs

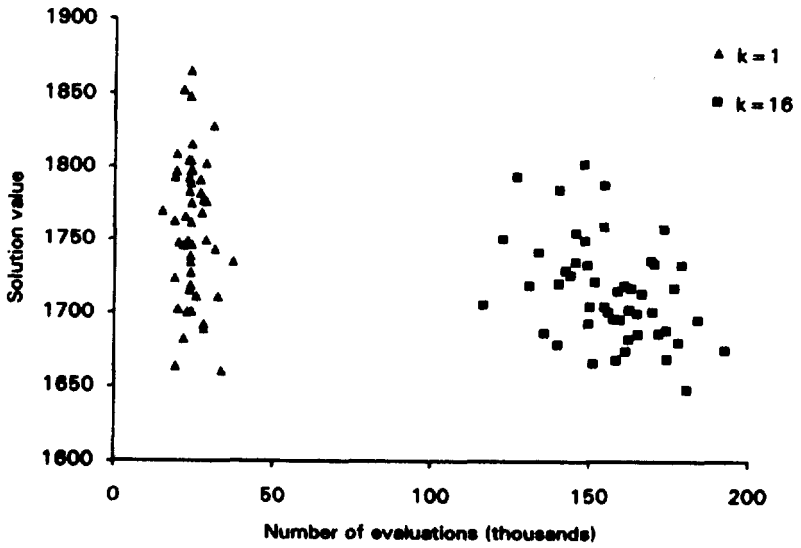

Fig. 4. Scatter plots for $k = 1$ and $k = 16$.

from different starting points which total to no more than this number of evaluations? This is the quantity which we need to estimate as a function of the allowed number of evaluations. In order to do this we propose to make a relatively large number of runs and then consider the results we would have obtained (in terms of the best solution so far at different stages) if these runs had occured in a different order. In this way we will make the best use of all the experimental data we obtain. In fact we will consider up to 2000 runs from different random starting points and then estimate the figure we want by taking the average of the best-so-far results from 200 different random reorderings of these runs. There are neater ways to make this estimate in the case that all runs are of the same, or nearly the same, length (see for example Johnson et al., 1989). In our experiments, however, the number of evaluations in a single run varied considerably depending on the starting point, as can be seen from Fig. 4.

The results obtained are shown in Fig. 5. Since changing the value of k has the effect of changing the number of evaluations in each run, we needed fewer runs with higher values of k. For every problem instance and parameter setting we made sufficient runs to generate 10 million solution evaluations. In each case the graph gives the average of results obtained on 10 different randomly generated instances of the problem at hand. The same problem instances were solved for each of three values of the spread parameter k. For each problem instance we record the best solution ever found (which is unlikely to be optimal for this size of problem). Then we look at the best solution found so far as a function of the number of evaluations for each of 200 different random reorderings of the data from individual runs of the local improvement method. These figures are then averaged for a given problem instance and recorded in terms of the percentage above the best solution found. The graphs show an average result from all 10 problem instances.

It is clear from Fig. 5 that, for this type of TSP, using the first-improving method is in general the best choice. We have carried out an extensive set of experiments with 50-city TSPs, for non-Euclidean as well as Euclidean distances. For these problems we also tried using the Or-opt move to define a neighbourhood – this takes a segment of 1, 2 or 3 cities from one position in the tour, and inserts them into a different place,

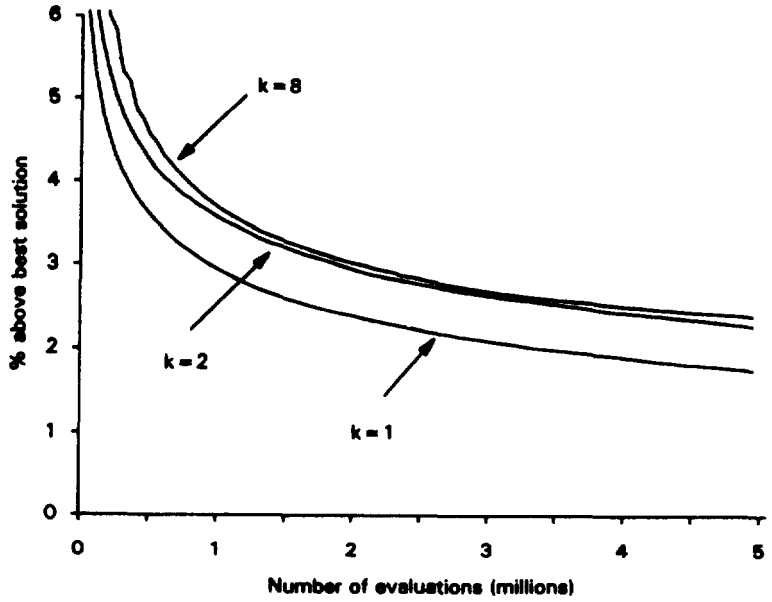

Fig. 5. Solution quality for TSP with 100 locations.

possibly reversed in sequence (see Golden and Stewart, 1985). All these experiments support the conclusion that first-improving is the method of choice. This is in agreement with the intuition we gained from considering the models of Section 3, given that we are concerned with both solution time and solution quality.

We also carried out experiments on the Quadratic Assignment Problem (QAP). In this case we generated both cost and flow matrices using uniformly distributed random numbers. Local search techniques have been found to be quite effective for QAP using a neighbourhood which simply swaps the assignment of two facilities between locations. It has been observed that a best-improving search is more effective than first-improving for QAP. This observation is complicated, however, by the fact that a complete search of the entire neighbourhood can be carried out more efficiently using the results from a complete search carried out on the previous step (see Burkard and Rendl, 1984). For this reason, the best-improving method, which requires a complete search of each neighbourhood, can obtain a time advantage over the first-improving method. In the experiments reported here this factor is ignored, and, as before, we just consider the total number of solutions evaluated.

Fig. 6 shows the results obtained for randomly generated 20-location QAPs, using the same approach as for the TSP problems (so that we consider 10 different problem instances and 200 different random reorderings of the local search results for each problem instance). As before the first-improving method appears to be best, but best-improving search is now seen to be competitive for problems in which a large amount of computation time is available. There is no *a priori* reason to suppose that the best choice of pivot rule should be insensitive to the amount of computing time available to solve a problem. We might expect the lines giving performance for different spread parameters to cross-over, with larger values of k beginning to dominate for very long computation times. We observed no evidence for this in practice for any of the TSP problems we looked at.

5. Predicting local search performance

It is natural to ask what are the characteristics of the problem, and in particular the characteris-

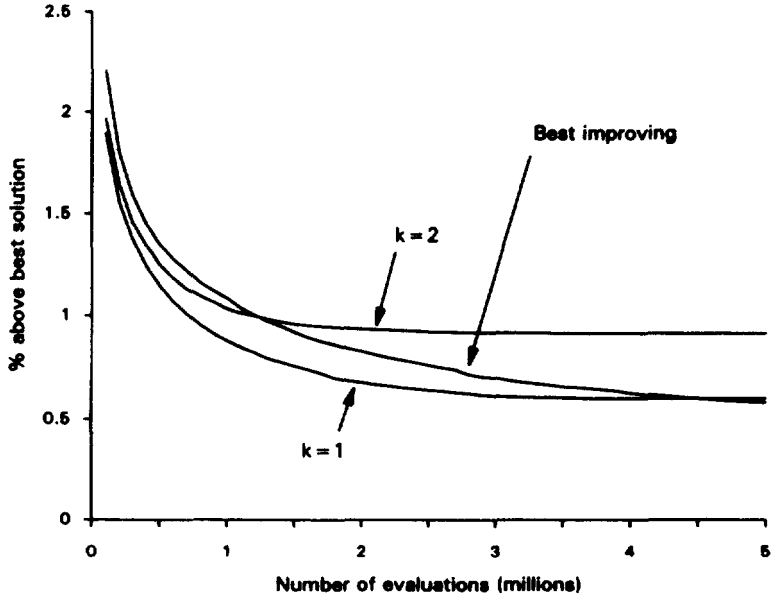

Fig. 6. Solution quality for QAP with 20 locations.

tics of the neighbourhood structure, which may cause one method to be better than another. One way to approach this question is to look at the structure of the neighbourhood graph. For example, it would be natural to expect local search techniques to work better on neighbourhood graphs with a relatively small diameter, or at least a small mean distance between nodes. A convenient proxy measure for this group of characteristics of the graph might be the second smallest eigenvalue of the difference Laplacian matrix (see Mohar, 1991). An alternative characteristic of the neighbourhood graph which has been suggested as relevant (Evans, 1987) is the size of the largest independent set of nodes (i.e. a set of nodes none of which are adjacent). This number gives a bound on the largest number of local optima which are possible.

In our view, however, the relationship between the neighbourhood structure and the objective function f is most important. This issue has been considered by other authors. Sorkin (1991) has an interesting discussion on the influence of what he calls the 'energy landscape' on the performance of simulated annealing. He argues that energy landscapes which are self-similar, or fractal, are those on which annealing will perform well. An alternative approach is to consider measures of the correlation between the values of neighbouring solutions (see for example Manderick et al., 1991).

In this section we will consider the average number of steps taken by a local search algorithm (or the average number of solutions considered). We will argue that these quantities give a measure of how effective local search will be for a particular problem and neighbourhood structure.

We begin by returning to the analysis of the problem with random assignments of values. In this case it is possible to carry out some calculations without knowing the distribution from which the values are drawn. First we need a straightforward lemma.

Lemma. *Suppose that the solution values are drawn from a continuous distribution. If $n \geqslant 1$ solutions have already been evaluated then the probability that the next solution evaluated is the best so far is* $1/(n+1)$.

Proof. The proof relies on the observation that each of the $s!$ different rank orderings that are possible between a sequence of s i.i.d. continuous random variables are equally likely. The assumption on the continuous distribution is necessary to ensure that there is zero probability of two values being the same. Suppose that we evaluate n different solutions and record their rank order. Then, given this rank ordering between the previously evaluated solutions there are $n+1$ equally likely rank orderings including the $(n+1)$st solution, corresponding to the $n+1$ positions in the rank order in which this solution can be inserted. Only one of these rank orderings has the last solution evaluated as the best, and so this occurs with probability $1/(n+1)$ as required. □

Using this result it is possible to calculate the expected number of evaluations for the setup of Theorem 1 using the first-improving algorithm. We write $E(n, h)$ for the expected number of further evaluations, given that we have made a number n so far and that the h most recent evaluations have not improved the current best solution, then $E(n, h)$ satisfies the recurrence relations below. First we define $K(n, h)$ to be the number of unevaluated neighbours that the current best solution had when it was first evaluated. Notice that in an infinite solution tree with each node having N neighbours, there will be N unevaluated neighbours of the first point looked at, but once a move has been made then there are only ever $N-1$ unevaluated neighbours. Thus $K(n, h) = N-1$ unless $n = h+1$, in which case $K(n, h) = N$.

Now we can write down the recurrence relations as follows:

$$E(n, h) = 1 + \frac{E(n+1, 0) + nE(n+1, h+1)}{n+1},$$

for $h < K(n, h)$, $n = 1, 2, \ldots$, and

$$E(n, K(n, h)) = 0.$$

The expected total number of solution evaluations is given by $1 + E(1, 0)$.

Though the complete set of recurrence relations are hard to solve exactly, they can be evaluated numerically for different values of N and this has been done in Table 1. It can be seen that

Table 1
Expected number of evaluations with random solution values and different neighbourhood sizes

N	Expected number of evaluations, M	Ratio M/N
5	8.70	1.74
10	17.32	1.73
50	88.25	1.76
100	177.26	1.77
1000	1780.17	1.78

the expected number of evaluations, M, is about 1.8 times the value of N for larger values of N. The neighbourhood graph considered here has no circuits, it is not hard to see that the expected number of evaluations with random function values on a real neighbourhood graph will be smaller (since even if solutions which have been looked at before are re-evaluated, the algorithm will not move to one of these solutions and this will reduce the effective neighbourhood size). The most closely related theoretical result in the literature is given by Tovey (1985) who shows that when the neighbourhood graph is given by a hypercube, the number of steps taken by a local search method is always less than $(3/2)eN$, regardless of the pivot rule used.

In practice, when function values are not random, the expected number of evaluations in a single run of a local search algorithm is much greater. The better behaved the function values, the more likely it is that the local search will take many steps before getting stuck in a local optimum. This will involve more evaluations in a single run; but, since this corresponds to better behaviour of the solution surface, the end result will be better, if many local search runs are carried out using in total a fixed amount of computation time. This is a rather different perspective than has been taken by other researchers in this area who have often seen it as preferable to have a small number of evaluations in a single run of a local search method. The way the figures work out in practice is illustrated by our experiments with a 50-city TSP. In this case the average number of evaluations per run for a Euclidean problem is $4.3N$; whereas for a non-Euclidean problem, with less good solution surface behaviour but the same neighbourhood structure, the average number of evaluations is $4.1N$.

We conjecture that the average number of evaluations per run, using a first-improving local search, can be used as a measure of the effectiveness of local search for a particular problem and neighbourhood structure. In particular in choosing between two neighbourhood structures which have the same neighbourhood size, we should prefer the neighbourhood which gives the greater number of evaluations per run.

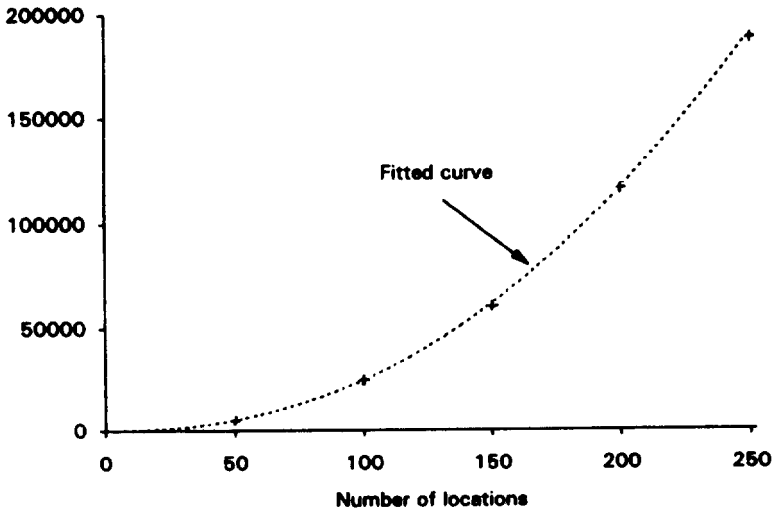

Fig. 7. Average number of evaluations in a single run for TSP.

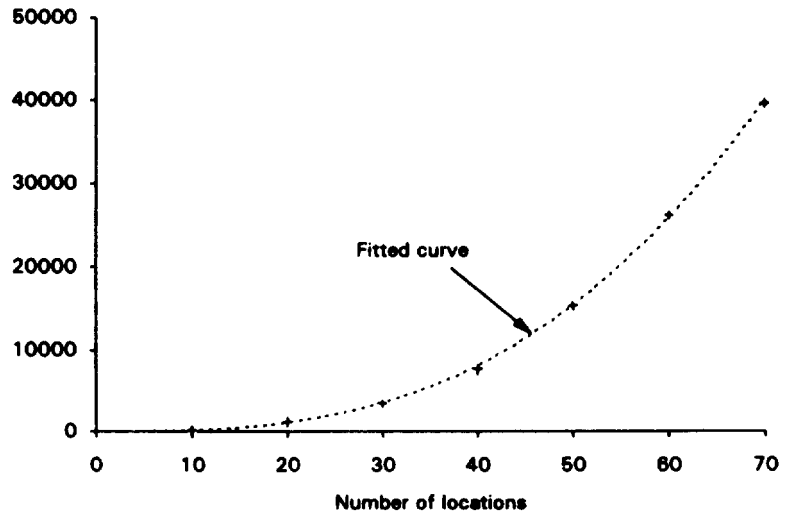

Fig. 8. Average number of evaluations in a single run for QAP.

It is also of interest to see how the number of evaluations per run increases as a function of the problem size. It is convenient to look at this as a function of the neighbourhood size N. Figs. 7 and 8 show how the number of evaluations changes as a function of problem size (rather than N) for both Euclidean TSP using the 2-opt neighbourhood and QAP using the swap neighbourhood. These results are averages from 100 different runs (using 5 different problems) at each problem size. In both cases fitted curves are shown which correspond to functions of the form αN^{β} for constants α and β. For TSP the number of evaluations is approximately $2.2(N^{1.1})$, and for QAP the number of evaluations is approximately $0.68(N^{1.4})$. It is also possible to look at the number of moves made in a local search run: for TSP this is approximately $1.5(N^{0.62})$ and for QAP it is approximately $0.68(N^{0.7})$.

Kern (1989) considers the use of the 2-opt neighbourhood for a Euclidean TSP in which points are placed according to a uniform distribution in the square. He shows that the expected number of evaluations in a single run is at most $O(n^{18})$ where n is the number of cities (i.e. $O(N^{9})$). There is thus a substantial difference between the empirical results we have observed and the best available theoretical bounds.

6. Conclusions

Our main result is that first-improving local search is superior to the other pivot rules we tried. This conclusion may not be surprising. It is in line with the intuitive reasoning given in Section 3. Moreover it is consistent with the experience of researchers who have looked at pivot rules for the simplex method (which can also be viewed as a local search procedure). It may be argued that for combinatorial optimization problems where local search is the method of choice, it is normally implemented in a more complex way, for example generating neighbours selectively in order to increase the chance of finding an improvement. It seems, however, that local search procedures are sufficiently important to make it worthwhile to fill in some of the gaps that may have been left, as it were, at the foundations.

We should point out that care is needed in reaching solid conclusions from experiments of the type we have made. There is almost no end to the computational experiments that one might wish to carry out in order to reach a firm recommendation on the best way to carry out local search. It is not surprising that most investigators have been more concerned to simply pursue the goal of the fastest code for the largest problem.

The most obvious shortcomings in our experimental work are that we have considered only small size problems and that we have not used the most effective neighbourhood structures for TSP (which are probably 3-Opt and that due to Lin and Kernighan, 1973).

We have also argued that recording the average number of evaluations in a single run of a local search method (or alternatively, the number of steps taken) will be useful in predicting the likely performance of local search on a particular problem class and with a particular neighbourhood structure. It may be thought counter-intuitive that we should prefer a neighbourhood structures in which a single local search run involves a greater amount of computation. More experimental work is needed to check whether the power of N relationships that we have observed also hold for larger problems.

References

Burkard, R.E., and Rendl, F. (1984), "A thermodynamically motivated simulation procedure for combinatorial optimization problems", *European Journal of Operational Research* 17, 169–174.

Evans, J.R. (1987), "Structural analysis of local search heuristics in combinatorial optimization", *Computers & Operations Research* 14, 465–477.

Golden, B.L., and Stewart, W.R. (1985), "The empirical analysis of heuristics", in: E. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D. Shmoys (eds.), *The Travelling Salesman Problem*, Wiley, New York.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1989), "Optimization by simulated annealing: An experimental evaluation; Part 1, graph partitioning", *Operations Research* 37, 865–892.

Kern, W. (1989), "A probabilistic analysis of the switching algorithm for the Euclidean TSP", *Mathematical Programming* 44, 213–219.

Lin, S., and Kernighan, B.W. (1973), "An effective heuristic algorithm for the travelling salesman problem", *Operations Research* 21, 498–516.

Manderick B., de Weger, M., and Spiessens, P. (1991), "The genetic algorithm and the structure of the fitness landscape", in: R.K. Belew and L.B. Booker (eds.), *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kauffman, San Mateo, CA.

Mohar, B. (1991), "Eigenvalues, diameter and mean distance in graphs", *Graphs and Combinatorics* 7, 53–64.

Papadimitriou, C.H., and Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NY.

Sorkin, G.B. (1991), "Efficient simulated annealing on fractal energy landscapes", *Algorithmica* 6, 367–418.

Tovey, C.A. (1985), "Hill climbing with multiple local optima", *SIAM Journal on Algebraic and Discrete Methods* 6, 384–393.

Wong, W.S., and Morris, R.J.T. (1989), "A new approach to choosing initial points in local search", *Information Processing Letters* 30, 67–72.

Yannakakis, M. (1990), "The analysis of local search problems and their heuristics", in: *Lecture Notes in Computer Science, Vol. 415*, Springer-Verlag, Berlin, 298–391.