SIMULATED ANNEALING AND COMBINATORIAL OPTIMIZATION*

Surendra Nahar **Sartaj Sahni** **Eugene Shragowitz**

University of Minnesota **University of Minnesota** **Control Data Corporation**

ABSTRACT

We formulate a class of adaptive heuristics for combinatorial optimization. Recently proposed methods such as simulated annealing, probabilistic hill climbing, and sequence heuristics, as well as classical perturbation methods are all members of this class of adaptive heuristics. We expose the issues involved in using an adaptive heuristic in general, and simulated annealing, probabilistic hill climbing, and sequence heuristics in particular. These issues are investigated experimentally.

Key words and phrases

Adaptive heuristics, Simulated annealing, Monte Carlo methods, optimization, perturbation functions, design automation

1. HEURISTIC METHODS

1.1. General Adaptive Heuristics

Simulated annealing is a general purpose combinatorial optimization technique that has been proposed by Kirkpatrick et al. [KIRK83]. This method is an extension of a Monte Carlo method developed by Metroplois et al., [METR53], to determine the equilibrium state of a collection of atoms at any given temperature T. Since the method was first proposed in [KIRK83], much research has been conducted on its use and properties. Some relevant references are: [BHAS85], [COHO83a], [COHO83b], [GOLD84], [JEPS83], [ARAG85], [KIRK83], [LUND83], [NAHA85], [ROME84ab], [SECH84], and [VECC83]. Most of the papers that report on an application of the method deal with problems that arise in the CAD area. This is not surprising as most CAD problems are known to be NP-complete, [SAHN80]. Hence, CAD problems are likely targets of solution by heuristic methods.

Simulated annealing as proposed by Kirkpatrick et al., [KIRK83], is a special case of a wider class of *adaptive heuristics* for combinatorial optimization. This wider class is formulated below. The term *adaptive*, in this context, simply means that some of the parameters of the heuristic may be modified. This modification may be done by the algorithm itself using some learning mechanism or may be done by the user using his/her own learning mechanism.

Consider any optimization problem. Suppose we wish to find a solution that minimizes the objective function $h()$ subject to certain constraints (a maximization problem may be solved by minimzing the negative of the objective function). Solutions that satisfy the constraints are called *feasible solutions* and a feasible solution with minimum $h()$ value is called an *optimal solution*. The *optimal value* of $h()$ is its minimum value. The general form of an adaptive heuristic to find a feasible solution with value close to optimal takes the form shown in Figure 1. The significance of the variables, functions, and procedures used in this algorithm are described below:

```
procedure GeneralAdaptiveHeuristic ;
  {General form of an adaptive heuristic for combinatorial
  optimization}
  S := S0; {initial solution}
  Initialize heuristic parameters;
  repeat
    repeat
      NewS := perturb (S );
      if accept (NewS , S ) then S := NewS ;
    until "time to adapt parameters";
    AdaptParameters ;
  until "terminating criterion";
end; {of GeneralAdaptiveHeuristic }
```

Figure 1 The general adaptive heuristic

S — This is the current solution to the optimization problem. Its initial value, S_0, may be any feasible solution. Generally, S_0 is either generated through some randomizing mechanism, i.e., a random feasible solution is used, or it is generated using some other heuristic for the problem being solved.

perturb — This is a function that generates a new feasible solution from the current solution S.

accept — This is a Boolean valued function that determines whether or not the new solution *NewS* should become the current working solution. *accept* is generally a function of the $h()$ (i.e., optimization function) values of S and *NewS* and of the heuristic parameters.

AdaptParameters — This procedure adapts the heuristic parameters. These parameters may include such things as the perturbation function, *perturb*, the acceptance function, *accept*, the current working solution, S, and even the criterion that determines when it is time to adapt the heuristic parameters.

The *generality of the general adaptive heuristic is readily seen from the observation that the specification of Figure 1 makes no mention of the specific problem to be solved.* To use the method, we need to be able to generate an initial feasible solution, be able to perturb a feasible solution and create another feasible solution, and be able to evaluate the objective function at these solutions. We show how several other heuristic classes may be obtained from the general adaptive heuristic of Figure 1.

*The research reported here was supported in part by the National Science Foundation under grants DCR-8305567 and DCR-8420935, and by Control Data Corporation under grant 84M301.

1.2. Classical Heuristics

Classical heuristics such as pairwise exchange, [SAHN85], are nonadaptive and accept a perturbation only if it improves the value of the objective function $h()$. These are trivially modeled by the general adaptive heuristic described above by making suitable selections for the heuristic parameters [NAHA85a]. For pairwise exchange perturbation function this results in the classical pairwise exchange heuristic of Figure 2.

```
procedure PairwiseExchangeHeuristic ;
  {General form of a pairwise exchange heuristic}
  S := S0; {initial solution}
  repeat
    NewS is obtained from S by making a pairwise exchange;
    if h(NewS) < h(S) then S := NewS;
  until S can be improved no further by a pairwise exchange;
end; {of PairwiseExchangeHeuristic }
```

Figure 2 A pairwise exchange heuristic

Using a pairwise exchange heuristic is quite straightforward as S_0 is the only unspecified parameter.

1.3. Simulated Annealing

The class of simulated annealing heuristics proposed by Kirkpatrick et al. [KIRK83], is obtained from the general adaptive heuristic of Figure 1 by making the following parameter selections:

(1) The acceptance function, *accept*, has the form:
if $(h(NewS) < h(S))$ **or** $(random < e^{(h(S)-h(NewS))/T})$
then *accept* := **true**
else *accept* := **false**;

where T is a heuristic parameter called "temperature" and *random* is a uniformly generated pseudo-random number in the range [0,1].

(2) The "time to adapt parameters" criterion is the number of iterations of the inner **repeat** loop that have been performed since the last adaptation.

(3) The procedure *AdaptParameters* does the following: The "temperature", T, used in the acceptance function is updated to $\alpha * T$ for some constant α, $0<\alpha<1$ and the number, *iterations*, of iterations of the inner **repeat** loop that are to be performed before the next adaptation is changed to β**iterations*. β is a constant that is at least 1.

(4) The "terminating criterion" is when we have used up the amount of computing time we wish to spend.

Substituting these selections into Figure 1, we obtain the form of the simulated annealing heuristic shown in Figure 3.

Simulated annealing is simpler to use than the general adaptive heuristic as there are fewer decisions to be made. We essentially need to determine the following:

(1) How is S_0 to be generated?

(2) What are the values of T_0; i_0; α, and β.

(3) How much computer time is the heuristic allowed?

These choices have a significant impact on the quality of (i.e., the value of S at termination) of the solution produced (see [NAHA85]). Determining optimal choices for these is not possible as the optimal choice depends not only on the particular problem being solved but also on the particular instance being solved. The time required to optimize the choices is, perhaps, better spent running the algorithm for a longer time.

The simplicity and generality of simulated annealing coupled with the development of convergence proofs ([LUND84] and [MITR85]) and the announcement of several promising experimental results ([GOLD84], [JEPS83], [KIRK83], [ROME84b], [SECH84], [VECC83]) have resulted in its widespread use in the design of algorithms for CAD applications. Indeed, hardware accelerators for simulated annealing have been constructed by major manufacturers of CAD tools. It is our view that *this effort is premature.* This is not because we do not believe that simulated annealing is a valuable contribution to the field of heuristic methods but because we believe that several issues surrounding it haven't, as yet, been fully studied.

```
procedure SimulatedAnnealing ;
  {General form of simulated annealing}
  S := S0; {initial solution}
  T := T0; {initial temperature}
  iterations := i0; {initial number of iterations of inner loop, ≥1}
  repeat
    repeat
      NewS := perturb(S);
      if (h(NewS)<h(S)) or (random < e^((h(S)-h(NewS))/T))
      then accept := true
      else accept := false;
    until inner loop has been repeated iterations times;
    T := α*T ; iterations := β*iterations
  until "out of time";
end; {of SimulatedAnnealing }
```

Figure 3 Simulated annealing

First and foremost, one must realize that *simulated annealing is just one instance of a more general class of adaptive heuristics.* Are there other instances that can be expected to perform better in practice? The fact that convergence to an optimal solution can be established under certain restrictive assumptions is of little practical value. Such proofs can be provided for several heuristic methods that one would not recommend for general use. For example, consider the random sampling heuristic of Figure 4. No mathematical sophistication is needed to realize that this does, in fact, converge to an optimal solution no matter which instance of which problem is being solved. The only requirement is that the procedure used to generate a random feasible solution be able to generate all feasible solutions.

```
procedure RandomSampling ;
  {Random sampling heuristic to determine a solution close
  to optimal}
  S := initial random feasible solution;
  repeat
    NewS := another random feasible solution;
    if h(NewS) < h(S) then S := NewS;
  until "out of time"
end; {of RandomSampling }
```

Figure 4 Random sampling heuristic

On the surface, this heuristic has several advantages over simulated annealing. It is much simpler than simulated annealing (we need only determine the amount of computer time available) and its convergence proof much easier. Despite these apparent advantages of the random sampling method, it is not a recommended heuristic for combinatorial optimization because *in practice, it does not perform well relative to other heuristics.*

While we can prove that both simulated annealing and random sampling converge to optimal solutions, we cannot tell when either has reached such a solution. Two general heuristic methods that overcome this difficulty are obtained by, respectively, adapting backtracking and branch-and-bound [HORO78]. The adaptation is quite straightforward as both these methods esentially search relevant parts of the solution space in a structured manner. In the adaptation, rather than terminate when the search is complete, we terminate when we have either used up the allotted time or the search is complete (whichever occurs first). Again, convergence is guaranteed by the fact that when enough time is provided, the algorithm is able to search (explicitly or implicitly) the entire solution space. Further, when this happens, the algo-

rithm is in a position to say that the current solution is indeed an optimal solution. Despite these positive aspects of the adapted backtracking and branch-and-bound heuristics, we are not in a position to recommend these methods as adequate empirical evidence regarding the rate of convergence is unavailable at this time. We propose to investigate this possibility in our future research.

Whether a heuristic is good or not depends on *its performance relative to other heuristics for the same problem* and *not on whether a convergence proof can be provided.* The *performance* of a heuristic must take into account *both the quality of the solutions produced and the amount of computational resources required.* If the required computational resources are not taken into account, then it is impossible to justify the acceptance of a suboptimal solution when an optimal one can be found in a finite (but large) amount of time.

Most of the early experimental results that were reported in the literature failed to compare simulated annealing with other heuristics that are known to be effective for the problems they are designed to solve. When such a comparison was made ([NAHA85], [ARAG85]), it was found that heuristics tailored to a specific problem usually obtained better solutions using only a fraction of the computing time used by simulated annealing.

Even though the results reported in [NAHA85] and [ARAG85] are negative vis-a-vis simulated annealing, this method cannot be totally discounted. This is because of its generality, relative simplicity, and its ability to outperform other simple and general heuristic methods (such as those of Figure 2 and 4). *Before simulated annealing can be recommended as a general purpose heuristic that is to be used in the absence of a clever tailored heuristic, one needs to address the issue of whether this is the best form of the general adaptive heuritic of Figure 1.*

1.4. Probabilistic Hill Climbing

The probabilistic hill climbing heuristics proposed by Romeo et al., [ROME84b], are obtained by generalizing simulated annealing to allow for the possibility of other temperature update methods, other functions than the exponential function used in *accept*, and other criteria to determine when the "inner loop termination criteria" and the temperature are to be changed. This generalization is given in Figure 5.

```
procedure ProbabilisticHillClimbing ;
  {General form of probabilistic hill climbing heuristics}
  S := S0; {initial solution}
  T := T0; {initial temperature}
  repeat
    repeat
      NewS := perturb (S);
      if (h (NewS)<h (S)) or (random < g (h (S), h (NewS),T)
        then accept := true
        else accept := false;
    until "time to terminate inner loop";
    update T and the inner loop termination criterion;
  until "out of time";
end; {of ProbabilisticHillClimbing }
```

Figure 5 Probabilistic hill climbing

As pointed out earlier, convergence proofs are nice but of little value in justifying a heuristic. For a heuristic, "the proof of the pudding is in the eating". The success stories are an indication that the simulated annealing approach of occassionally accepting bad perturbations (i.e., ones with higher objective function value) has some merit. There are, however, a large number of ways in which this can be accomplished. Nahar, Sahni, and Shragowitz, [NAHA85], studied 20 different acceptance functions and discovered that many of these often outperform the simulated annealing acceptance function. The studies reported in [NAHA85] lead us to formulate yet another specialization of the class of general adaptive heuristics. This class is called the "sequence oriented heuristics" and in described in the next section.

1.5. Sequence Heuristics

The best results obtained in [NAHA85] used the following strategy: accept a new solution $NewS$ with $h(NewS) \geq h(S)$ iff the last k perturbations on S failed to generate a $NewS$ with $h(NewS) < h(S)$. So, bad perturbations are accepted only if we have been unable (over a sequence of attempts) to find a good perturbation. This method may be generalized to obtain the algorithm of Figure 6.

```
procedure SequenceHeuristic ;
  {Sequence oriented heuristic}
  S := S0; {initial solution}
  L := L0; {initial sequence length}
  length := 0; {current length of bad perturbation sequence}
  repeat
    repeat
      NewS := perturb (S);
      if h (NewS) < h (S) then [S := NewS; length := 0;]
                          else [length := length + 1];
    until length > L ;
    UpdateLength ;
    UpdateS ;
  until "terminating criterion";
end; {of SequenceHeuristic }
```

Figure 6 Sequence heuristic

In the sequence heuristic used in [NAHA85], the procedure *UpdateS* keeps track of the best solution found so far and also updates S to $perturb(S)$. The procedure *UpdateLength* is a null procedure (i.e., it does nothing). In a more general application of the sequence heuristic, *UpdateLength* could increase the value of L to (say) $L+\beta$ or $\beta * L$.

1.6. Issues

There are several issues that must be resolved before the general adaptive heuristic or one of its adaptive specializations (i.e., simulated annealing, probabilistic hill climbing, or sequence heuristics) may be used. In view of the large amounts of computing time being spent to obtain solutions via simulated annealing, the possiblity of incorporating some learning into the adaptive mechanisms of these heuristics appears to be worth studying. Support is lent to this recommendation by the results of [NAHA85] which show that apriori determination of the constants α and β in simulated annealing and the restriction to the single $g()$ function of simulated annealing does not lead to an efficient implementation. Further, experiments with near optimal implementations using different $g()$ functions showed that on any given instance any of the studied $g()$ functions could produce the best results. We shall not study the learning issue here. The issues we shall examine are listed below.

(1) **Adaptive heuristic or tailored heuristic**
Are adaptive heuristics a substitute for well thought out heuristics that are tailored to the specific problem to be solved? Our experiments show that *custom heuristics often generate better solutions than those generated by adaptive heuristics and do so using a fraction of the computing time.*

(2) **Choosing S_0**
We study the effect of selecting S_0 by some random mechanism versus using a solution generated by some "clever" heuristic. Our experiments indicate that *better solutions* are obtained when one starts with a good S_0.

(3) **Choice of the perturbation function**
One might expect that the performance of an adaptive heuristic will be affected by the perturbation function that is used. Since there appears to be no experimental validation of this thesis, we conducted such a study. Our studies confirm this thesis.

(4) **Simulated annealing vs sequence heuristic**
The sequence heuristic is slightly easier to use than simulated annealing as one fewer parameter needs to be selected. It is also more elegant as it does not contain the "artificial" notion of a temperature. The results reported in [NAHA85] indicate that it is superior to simulated annealing. We provide addtional evidence to support this claim.

2. PERTURBATION FUNCTIONS

Our experiments were confined to the three problems: optimal linear arrangement, traveling salesman, and circuit partitioning. These are described in later sections. It is sufficient to know that the first two of these require us to find an optimal permutation of n objects while the last requires us to optimally partition a set of n objects into two subsets. The perturbation functions we experimented with are described below for the two categories of problems we experimented with. For a permutation problem, the current solution S is described by a permutation, σ, of the n objects. The current solution, S, for a partition problem is described by the two partitions A and B.

1. Single Move (SM)

For a permutation problem an object is selected randomly and moved to a random location in σ. For example, if $\sigma =$ (3, 5, 1, 2, 4) and we select object 2 and move it to position 1, we get the permutation (2, 3, 5, 1, 4). In the case of a partition problem, the single move perturbation requires us to select one of the n objects and move it from its current partition to the other partition.

2. Pairwise Exchange (PE)

In the case of a permutation problem, σ is perturbed by interchanging the positions of two randomly chosen objects. In the case of a partition problem, two elements are selected at random. One from A and the other from B. The element selected from A is moved to B and that selected from B is moved to A.

3. Cycle of Length 3 (C3)

This perturbation strategy is used only for permutation problems. Three objects are chosen at random. Let i, j, and k be the three objects selected. i is moved to the position in σ currently occupied by j; j to the position occupied by k; and k to that formerly occupied by i. As can be seen, this is a natural extension of pairwise exchange to the case of three elements.

4. Two Bond Exchange (TBE)

This is the strategy used in the Lin-Kernighan heuristic for the traveling salesman problem [LIN73]. We use this on permutation problems only. Let $(s_1, s_2, \cdots, s_n)$ be the current permutation σ. We say there is a bond between s_1 and s_2, s_2 and s_3, ..., s_{n-1} and s_n, and s_n and s_1 in the current permutation. In the two bond exchange perturbation strategy, two bonds are chosen at random and broken. These are replaced by the two unique bonds required to rejoin the permutation (and create a new one). For example, if the bonds (s_i, s_{i+1}) and (s_j, s_{j+1}) are broken, for some i and j, $i < j < n$, the new bonds are (s_i, s_j) and (s_{i+1}, s_{j+1}). The net result is that the permutation segment between s_{i+1} and s_j is reversed. The perturbed permutation is: $(s_1, \ldots, s_i, s_j, s_{j-1}, \ldots, s_{i+1}, s_{j+1}, \ldots, s_n)$.

5. Pair + Single Move (P+SM)

This is used by us on subset problems alone. First, a randomly selected object from subset A is exchanged with a randomly selected object from subset B (i.e., a pairwise exchange is performed). Then, an object is selected from one of A and B and moved to the other subset (i.e., a single move is performed).

6. Pair + Pair (P+P)

As in the case of the P+SM heuristic, the P+P heuristic is used on subset problems alone. In this heuristic, a randomly selected pair of objects (one from A and the other from B) are interchanged. This is done once again to get the perturbed solution.

3. EXPERIMENTAL RESULTS

3.1. OPTIMAL LINEAR ARRANGEMENT (OLA)

In the optimal linear arrangement (OLA) problem, we are given n circuit elements (cells, boards, chips, etc) and a set of nets which interconnect the circuit elements. For any linear ordering of these n elements, the maximum number of nets that cross between any pair of adjacent elements is called the *density* of the linear arrangement. We are required to find a linear ordering with minimum density. This problem is identical to the board permutation problem studied in [GOTO77], and [COHO83a]. The problem also arises in the placement of standard cells and gate arrays [KANG83], and in the ordering of via columns in single row routing [RAGH84] and [TING78].

We generated 10 random instances of the optimal linear arrangement problem. Each instance consisted of 50 circuit elements and 500 two point nets. For each of the 10 instances, a starting solution S_0 was obtained by generating a random permutation. The sum of the densities of these 10 permutations was 2516. The greedy heuristic proposed by Goto, [GOTO77], was first used to obtain linear arrangements for the 10 instances. The greedy heuristic of [GOTO77] required approximately 18 seconds per instance (all experiments reported in this paper were conducted on an Apollo DN320 workstation, all programs were coded in Pascal) and produced solutions whose density summed to 658 less than that for the 10 random solutions.

We experimented with three parameter sets for simulated annealing. In all cases, the temperature update constant α was set to 0.95. The three parameter sets are described below:

Parameter Set 1 (PS1)
$T_0 = 10$, initial number of iterations of inner **repeat** loop = 30, multiplicative change factor $\beta = 1.02$. Using the above α of 0.95, the temperature reduced to 0.87 (on the average) when the heuristic terminated.
PS2
$T_0 = 10$, initial number of iterations of inner **repeat** loop = 150, multiplicative change factor $\beta = 1.02$. The average value of the temperature at termination was 4.63.
PS3
$T_0 = 2.5$, initial number of iterations of inner **repeat** loop = 150, multiplicative change factor $\beta = 1.02$. At the time of termination, T had the value 1.16 (on the average).

Figure 7 gives the reduction in the sum of densities accomplished by each of the three variations of simulated annealing described above using each of the four perturbation strategies described earlier for permutation problems. The times given in the TIME column are times per instance.

Observations

(1) It is striking to observe that *none of the 12 simulated annealing strategies could match the performance of Goto's greedy heuristic* (658).

(2) Further, as anticipated, the choice of the perturbation function has a significant effect on the peformance of simulated annealing.

(3) As pointed out in other studies (e.g., [NAHA85]), the initial temperature and the temperature update constant α have a significant impact on the performance of simulated annealing.

For the sequence method, three parameter sets were also used. In all cases, *UpdateLength* sets *length* to β*length* (early experiments with β+*length* revealed this to be inferior to β*length*). The three parameter sets are characterized below:

PS1

$L_0 = 150$, $\beta = 1$ (in *UpdateLength*), and *UpdateS* updates S to *perturb* (S).

PS2

$L_0 = 150$, $\beta = 1$, and *UpdateS* updates S to a random permutation rather than one obtained by using the current permutation strategy on S.

PS3

$L_0 = 150$, $\beta = 1.02$, and *UpdateS* updates S to *perturb* (S). At the time of termination, the sequence length had increased to 175 on the average.

Figure 8 gives the reduction in the sum of densities accomplished by each of the three variations of the sequence method described above using each of the four perturbation strategies described earlier for permutation problems.

TIME	PS1				PS2				PS3			
sec	SM	PE	C3	TBE	SM	PE	C3	TBE	SM	PE	C3	TBE
180	271	298	369	320	274	324	303	280	451	485	477	338
360	363	392	430	372	308	355	354	308	494	556	531	394
540	448	506	496	403	312	370	390	336	549	590	561	406
720	511	559	553	430	328	374	402	341	557	596	571	407
900	547	589	576	431	350	397	413	358	576	613	593	430

PSi = parameter set i, SM = single move, PE = pairwise exchange, C3 = cycle of length 3, TBE = two bond exchange

Figure 7 OLA density reductions for simulated annealing

Observations

(1) The improvements obtained by the sequencing method are clearly superior to those obtained by simulated annealing.

(2) In the case of the OLA problem, simulated annealing is not effective in obtaining improved solutions.

(3) The perturbation function with best performance for the sequence method is pairwise exchange. However, when we started from a random solution, the cycle of length 3 perturbation scheme was best.

TIME	PS1				PS2				PS3			
sec	SM	PE	C3	TBE	SM	PE	C3	TBE	SM	PE	C3	TBE
180	440	531	537	312	414	525	552	329	441	537	552	295
360	488	589	575	402	451	566	580	362	475	581	577	404
540	512	608	598	444	492	571	585	373	491	601	623	420
720	540	618	620	458	525	574	605	396	502	631	632	483
900	553	633	637	469	527	587	612	396	503	642	646	483

Figure 8 OLA density reductions for the sequence method

Observations

Observations 1 and 2 from the simulated annealing experiment apply here too. However, the sequence method using parameter set 3 and the perturbation methods pairwise exchange (PE) and cycle of length 3 (C3) came very close to matching the performance of Goto's heuristic. Of course, this is done at the expense of using 50 times the computing time. For all three parameter sets the cycle of length 3 perturbation method produced the best results.

Since neither simulated annealing nor the sequence method are able to outperform Goto's heuristic, we may wonder if these heurictics can obtain any improvement when the starting solution S_0 is that produced by Goto's heuristic. Figures 9 and 10 give the total improvements obtained from this starting solution.

TIME	PS1				PS2				PS3			
sec	SM	PE	C3	TBE	SM	PE	C3	TBE	SM	PE	C3	TBE
180	0	0	0	4	0	0	0	0	3	0	0	7
360	0	0	0	4	0	0	0	0	3	2	0	7
540	0	0	0	4	0	0	0	0	3	2	0	7
720	0	2	0	4	0	0	0	0	3	3	0	7
900	0	5	0	4	0	0	0	0	3	3	2	7

Figure 9 OLA density reductions for simulated annealing beginning with Goto's solution

3.2. CIRCUIT PARTITION

The cell partition problem is one of the problems reported in [KIRK83]. The input to this problem is quite similar to that for NOLA. The essential difference is that each circuit element (called a cell) has a size associated with it. The cells are to be partitioned into two groups A and B. Let *Nets* be the number of nets that are in both A and B. Let $\Delta Size$ denote the magnitude of the difference is size between A and B. The objective is to find a partition (A,B) that minimizes $r(A,B) = Nets + \Delta Size$. In this measure, *Nets* and $\Delta Size$ are weighted equally. Kirkpatrick et al. assigned different weights to these two components.

As a bench mark heuristic, we used the partitioning heuristic of Lin [KERN70]. This was modified to account for the area of circuit elements directly in the objective function rather than by replacing each element by many (as suggested in [KERN70]). 10 instances, each having 50 elements and 500 nets were used. These were randomly generated. The cost of the 10 random starting solutions for these instances was 2974. The modified heuristic ([KERN70]) obtained a cost reduction of 1100 and required an average of 3.3 seconds per instance.

For each of the three simulated annealing variations used, we set α (the temperature update constant) to 0.98. The remaining characteristics were:

PS1

$T_0 = 40$, initial number of iterations of inner **repeat** loop = 30, multiplicative change factor $\beta = 1.02$. The average final value of T was 0.69 for this parameter set.

TIME	PS1				PS2				PS3			
sec	SM	PE	C3	TBE	SM	PE	C3	TBE	SM	PE	C3	TBE
180	0	9	1	20	0	1	3	13	0	14	11	14
360	4	26	8	20	0	2	7	13	0	20	13	16
540	4	27	12	20	0	4	7	13	0	29	15	19
720	4	28	15	20	0	12	7	13	0	38	17	19
900	4	29	20	20	0	12	7	13	0	40	20	19

Figure 10 OLA density reductions for the sequence method beginning with Goto's solution

PS2
$T_0 = 40$, initial number of iterations of inner **repeat** loop = 150, multiplicative change factor $\beta = 1.02$. For this parameter set, the average final value of T was 3.0.
PS3
$T_0 = 15$, initial number of iterations of inner **repeat** loop = 150, multiplicative change factor $\beta = 1.02$. In this case, the average final value of T was 1.15.

The cost reductions obtained by simulated annealing when S_0 is a randomly generated partition are given in Figure 11.

TIME	PS1				PS2				PS3			
sec	SM	PE	PE+SM	P+P	SM	PE	PE+SM	P+P	SM	PE	PE+SM	P+P
180	840	980	938	920	853	870	846	844	846	940	897	904
360	882	1050	987	923	884	950	920	870	857	995	940	921
540	898	1051	1014	923	894	1004	932	907	879	1037	949	921
720	899	1057	1017	923	905	1019	945	915	881	1037	977	921
900	910	1057	1024	923	920	1032	954	915	897	1037	977	921

Figure 11 Circuit partition cost reductions for 10 fifty circuit instances using simulated annealing

Observations

(1) Despite the fact that each of the twelve simulated annealing strategies was provided almost 300 times as much time as required by the Lin-Kernighan heuristic, none was able to produce solutions comparable to those generated by this greedy heuristic! PS1-PE was the only combination that came within 5% of the greedy heuristic solutions.

(2) As in the case of our other tests, the choice of the perturbation function has a material affect on the quality of the solutions generated.

(3) Unlike our other tests, the parameter set did not have any significant effect on the quality of the solutions generated.

The three parameter sets used with the sequence method are described below:

PS1
$L_0 = 150$, $\beta = 1$ (in *UpdateLength*), and *UpdateS* updates S to *perturb* (S).
PS2
$L_0 = 150$, $\beta = 1$, and *UpdateS* updates S to a random permutation rather than one obtained by using the current permutation strategy on S.
PS3
$L_0 = 150$, $\beta = 1.02$, and *UpdateS* updates S to *perturb* (S). The average final value of the sequence length was 2000.

Figure 12 gives the cost reductions obtained when the sequence method is used and S_0 is a randomly generated partition.

Observations

(1) As in the case of simulated annealing, the sequence method was unable to match the performance of the Kernighan-Lin heuristic. Unlike our tests with the other two problems, simulated annealing outperformed the sequence method.

(2) Both the parameter set and the perturbation function have some effect on the quality of the solutions produced. This effect is less noticeable as the algorithms are run for more and more time.

When simulated annealing and the sequence method are started with the solution produced by the Kernighan-Lin greedy heuristic, *virtually no improvement was observed.*

4. CONCLUSIONS

We have formulated a class of general adaptive heuristics and have pointed out the possibility of introducing learning capabilities into heuristics. Some of the additional issues related to the use of general adaptive heuristics have been pointed out. A thorough empirical study of the performance of simulated annealing relative to that of the sequence method was carried out. Both methods are similar in that they both make random perturbations and both accept good as well as bad perturbations. In addition, a comparison with heuristics designed to solve a specific problem was also conducted. With the respect to the issues raised in Section 1.6, our conclusions are:

(1) *A general randomization heuristic (such as simulated annealing or the sequence method) is not a substitute for a well designed heuristic that is tailored to the specific problem being solved.* However, by extrapolating our experimental results, we expect that the randomization heuristics would have matched or exceeded the performance of the tailored heuristics if sufficiently more time was provided.

(2) *For some problems, a good starting solution may yield better results than a random starting solution.* Once again, we expect that if the randomization heuristics are run for a longer time than allowed in our tests, then random starting solutions will produce as good solutions as good starting solutions.

(3) *Selecting the appropriate perturbation function appears to be very important.*

TIME	PS1				PS2				PS3			
sec	SM	PE	PE+SM	P+P	SM	PE	PE+SM	P+P	SM	PE	PE+SM	P+P
180	699	901	829	804	727	876	825	849	698	945	840	847
360	804	955	871	863	777	906	892	870	746	983	891	903
540	829	955	880	884	795	932	925	880	765	998	909	916
720	851	964	897	897	806	943	936	898	794	1012	913	927
900	898	972	907	909	807	967	936	901	810	1014	962	930

Figure 12 Circuit partition cost reductions for 10 fifty circuit instances using the sequence method.

(4) The choice of T_0,α, and β (cf. simulated annealing) are important when computational resources are limited and less crucial when this is not the case. The choice of β and L_0 are important for the sequence method.

(5) *In our tests, the sequence method performed marginally better than did simulated annealing. This adds support to our view that the observed performance of simulated annealing on combinatorial problems is due largely to its acceptance of bad perturbations in order to escape from local minima rather than to some mystical connection between combinatorial problems and the annealing of metals.*

Experiments were also performed using the Traveling Salesman problem. Results may be found in [NAHA85a].

5. REFERENCES

[ARAG85] C. Aragon, D. Johnson, L. MeGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation.

[BHAS85] J. Bhasker and S. Sahni, Optimal linear arrangement of circuit components, University of Minnesota, Technical Report, 1985.

[COHO83a] J. Cohoon and S. Sahni, Heuristics for the board permutation problem, *Proceedings 1983 IC CAD Conference*, Sept 1983.

[COHO83b] J. Cohoon and S. Sahni, Heuristics for the circuit realization problem, *Proceedings 20th Design Automation Conference*, June 1983.

[GEMA83] D. Geman and S. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.

[GOLD84] B. Golden and C. Skiscim, Using simulated annealing to solve routing and location problems, University of Maryland, College of Business Administration, Technical Report, Jan. 1984.

[GOTO77] S. Goto, I. Cederbaum, and B.S. Ting, "Suboptimal Solution of the Backboard Ordering with Channel Capacity Constraint", *IEEE Trans. Circuits and Systems*, Nov. 1977, pp. 645-652.

[GREE84] J. Greene and K. Supowit, Simulated annealing without rejected moves, Proceedings ICCD, Oct. 1984, pp 658-663.

[HORO78] E. Horowitz and S. Sahni, Fundamentals of computer algorithms, Computer Science Press, 1978.

[JEPS83] D. Jepsen and C. Gelatt Jr., Macro placement by Monte Carlo annealing, *Proceedings 1983 IC CAD Conference*, Sept 1983, pp. 495-498.

[KANG83] S. Kang, Linear ordering and application to placement, *Proceedings 20th Design Automation Conference*, 1983, pp 457-464.

[KERN70] B. Kernighan and S. Lin, An efficient heuristic procedure for the partitioning of graphs, Bell Systems Tech. Jr., Feb. 1970.

[KIRK83] S. Kirkpatrick, C. Gelatt, Jr., and M. Vecchi, Optimization by simulated annealing, *Science*, Vol 220, No 4598, May 1983, pp. 671-680.

[LIN73] S. Lin and B. Kernighan, An effective heuristic for the traveling salesman problem, *Operations Research*, Vol 21, pp. 498-516, 1973.

[LUND83] M. Lundy and A. Mees, Convergence of the annealing algorithm, University of Cambridge, 1983.

[METR53] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller, Equation of state calculations by fast computing machines, *Jr. Chem. Phys.*, Vol 21, p. 1087, 1953.

[MITR85] D. Mitra, and F. Romeo, and A. Sangiovanni-Vincentelli, Convergence and finite-time behavior of simulated annealing, Technical Report UCB/ERL M85/23, University of California, Berkeley, March 1985.

[NAHA85] S. Nahar, S. Sahni, and E. Shragowitz, Experiments with simulated annealing, *22nd Design Automation Conference*, 1985, pp. 748-752.

[NAHA85a] S. Nahar ,S. Sahni and E. Shragowitz, Simulated Annealing and Combinatorial Optimization, University of Minnesota, Minneapolis, Technical report # 85-56, 1985.

[RAGH84] R. Raghavan and S. Sahni, The complexity of single row routing, *IEEE Trans. On Circuits and Systems*, Vol CAS-31, No 5, May 1984, pp. 462-471.

[ROME84a] F. Romeo, A. Vincentelli, and C. Sechen, Research on simulated annealing at Berkeley, Proceedings ICCD, Oct. 1984, pp 652-657.

[ROME84b] F. Romeo and A. Vincentelli, Probabilistic hill climbing algorithms: Properties and applications, University of California, Berkeley, UCB/ERL M84/34, 1984.

[SAHN80] S. Sahni and A. Bhatt, Complexity of design automation problems, *Proceedings 17th Design Automation Conference*, 1980, pp. 402-411.

[SAHN85] S. Sahni, Software development in Pascal, Camelot Publishing Co., Fridley, Minnesota, 1985.

[SECH84] C. Sechen and A. Sangiovanni-Vincentelli, The Timberwolf placement and routing package, 1984.

[STEW77] W. Stewart, A computationally efficient heuristic for the travelling salesman problem, *Proceedings of the 13th Annual Meeting of Southeastern TIMS*, Myrtle Beach, S.C., pp 75-83, 1977.

[TING78] B. Ting and E. Kuh, An approach to the routing of multilayer printed circuit boards, *Proc. IEEE Symp. On Circuits and Systems*, pp. 902-911, 1978.

[WHIT84] S. White, Concepts of scale in simulated annealing, Proceedings ICCD, Oct 1984, pp 646-651.

[VECC83] M. Vecchi and S. Kirkpatrick, Global wiring by simulated annealing, *IEEE Trans. On Computer Aided Design*, Vol CAD-2, No 4, Oct. 1983, pp 215-222.