A Comparative Study of Genetic Search

Kihong Park
Computer Science Department
Boston University
Boston, MA 02215

Abstract

We present a comparative study of genetic algorithms and their search properties when treated as a combinatorial optimization technique. This is done in the context of the NP-hard problem MAX-SAT, the comparison being relative to the Metropolis process, and by extension, simulated annealing. Our contribution is two-fold. First, we show that for *large* and *difficult* MAX-SAT instances, the contribution of cross-over to the search process is marginal. Little is lost if it is dispensed altogether, running mutation and selection as an enlarged Metropolis process. Second, we show that for these problem instances, genetic search consistently performs worse than simulated annealing when subject to similar resource bounds. The correspondence between the two algorithms is made more precise via a *decomposition argument*, and provides a framework for interpreting our results.

1 Introduction

Genetic algorithms [11] are increasingly being viewed as general-purpose optimization techniques applicable to a wide range of problem domains [5, 6, 12, 13, 15, 21]. The combination of the "survival of the fittest" principle and pairwise interaction among members of a population yields a non-local, yet intuitively appealing means of generating new elements, complementing the local mechanism facilitated by mutation. As a model of physical processes, genetic algorithms and its mechanics are interesting in their own right, irrespective of how "well" the system performs its function. For example, in the field of genetic programming, it may be of more relevance that the programming task can be automated to a certain degree than being able to say that the evolved program is optimal, or that the process by which it was arrived at was more efficient than some other. Nonetheless, the notion of fitness function is ingrained in the basic formulation of genetic algorithms, and understanding a GA's prowess at finding optimal solutions is of indirect interest even when the motivation behind the problem under study is not one of combinatorial optimization.

When treated as a strict optimization technique, the focus shifts exclusively to "how well" genetic algorithms fair relative to other algorithms. Although research abounds [3, 10, 12, 13, 15, 17, 21], the jury is still out with respect to how effective GAs are at solving hard combinatorial optimization problems. In part, this is due to the nonuniformity of problem instances which makes comparing results across different domains difficult. In other cases, the problem sizes are too small to render the reported results typical, the problem instances are selected without specific regard to their combinatorial "hardness," and performance evaluations relative to other algorithms under comparable resource bounds are not explicated.

In this paper, we investigate the search properties of genetic algorithms in the context of MAX-kSAT, a NP-hard combinatorial optimization problem. MAX-kSAT is the optimization version of the corresponding decision problem kSAT, where given a Boolean formula in conjunctive normal form having n variables (or literals), m clauses, and k variables per clause, an assignment is sought that maximizes the number of true clauses[1]. Hence, if the formula is satisfiable, an optimal assignment yields the *fitness value* m. In the following, we will work exclusively with MAX-3SAT, a popular variant. This paper extends previous work on the MAX-CLIQUE problem [4, 17] where it is shown that on a set of large, difficult problem instances belonging to the DIMACS benchmark set, simulated annealing consistently outperforms genetic algorithms. Although both problems are NP-hard, there is a significant difference with respect to their approximability. MAX-kSAT has an approximation algorithm with

[1] Note, although 2SAT $\in$ P, MAX-2SAT is NP-hard. In fact, it is *complete* for a large class of problems called MAX-SNP under a reduction that preserves approximability.

worst-case ratio 3/4 (for 3SAT it is even better at 7/8) whereas for MAX-CLIQUE, no constant-ratio approximation scheme is known [22]. Moreover, recent work has shown that unless P = NP, MAX-CLIQUE cannot be feasibly approximated to within a polynomial factor, much less a constant one [2]. In the case of MAX-kSAT, the existence of a bound on the constant approximation factor can be shown, thus making it unlikely that arbitrarily close approximations can be feasibly achieved [1]. The aforementioned facts provide strong evidence to indicate that MAX-kSAT is an inherently "easier" problem than MAX-CLIQUE, and it is interesting to ask whether the results obtained in [4, 17] are applicable to this problem domain. For instance, "easiness" may imply that the *building-block hypothesis* [9] is a more fitting attribute of instances belonging to this problem set, giving genetic algorithms an edge by virtue of its cross-over operator which is conjectured to exploit this feature. In the following sections, we will show that for large, difficult (see the definition below) 3SAT instances, cross-over contributes only marginally to the search process, and genetic search exhibits performance consistently worse than simulated annealing given similar resource bounds.

To select *generically* difficult 3SAT instances without exerting a priori bias toward genetic search or simulated annealing, we will adopt the scheme proposed in [14], where it is shown that the difficulty of 3SAT instances, as reflected by the amount of work done by the David-Putnam procedure in determining the satisfiability of a Boolean formula, varies as a function of the ratio $\rho = m/n$ (number of clauses / number of variables), exhibiting a sharp "phase transition" at $\rho \approx 4.25$. For $\rho < 4.25$, problem instances are *underconstrained*, and not only are they satisfiable with high probability, but there is an abundance of satisfying assignments. For $\rho > 4.25$, a formula is *overconstrained*, and thus unlikely to be satisfiable. The difficult instances for the decision problem are shown to be concentrated at $\rho \approx 4.25$, and this will be our main source for benchmark instances.

The second goal of the paper is to provide a framework for understanding the performance results. We follow the approach of [17] where a *decomposition* method was proposed for isolating and evaluating the contribution of cross-over to the search process of a genetic algorithm. The gist of the idea lies in separating the three components of a GA, *cross-over* ($\mathcal{C}$), *mutation* ($\mathcal{M}$), and *selection* ($\mathcal{S}$), comparing the performance of the full algorithm denoted $\mathcal{CMS}$ (cross-over, mutation, selection), and the partial version consisting of cross-over and selection ($\mathcal{CS}$), against $\mathcal{MS}$, the version that uses only mutation and selection. By doing so, the synergistic as well as individual effects of cross-over are discerned. Noting that $\mathcal{MS}$ is just a Metropolis process (simulated annealing at constant temperature) "in disguise," a meaningful comparison between genetic search and simulated annealing, via extension from the Metropolis process, can be established.

This paper is organized as follows. In the next section we will describe the decomposition method in more detail, showing in what sense $\mathcal{MS}$ and the Metropolis process are related. This is followed by experimental results which evaluate the contribution of cross-over to the search process, and measure the performance of genetic algorithms and simulated annealing on MAX-3SAT. Augmenting that are sections describing the effects of population size, policies for maintaining diversity, and the evaluation of cross-over on a hybrid algorithm that combines genetic algorithms and simulated annealing. We conclude with a discussion of the present methodology and some implications.

2 Characterization of Genetic Search

2.1 Isolating Cross-over's Contribution

Evaluating the effects of cross-over in the overall search process of a GA has been an important theme [7, 8, 19, 20]. Given the strict combinatorial optimization setting of this paper, we take a direct approach by comparing the performance of $\mathcal{CMS}$ and $\mathcal{CS}$ against $\mathcal{MS}$ which does not employ the cross-over operator. Let p_c and p_m denote the cross-over rate and mutation rate, respectively. Viewing genetic search, $\mathcal{CMS}_{<p_c,p_m>}$, as parameterized by these two variables, we have

$$\mathcal{CS}_{p_c} \equiv \mathcal{CMS}_{<p_c,0>}, \quad \mathcal{MS}_{p_m} \equiv \mathcal{CMS}_{<0,p_m>}.$$

When $\mathcal{CMS}$, $\mathcal{CS}$, and $\mathcal{MS}$ are used with the subscripts omitted, it is done with the understanding that the best observed performance over a wide range of p_c and p_m values is being referred to. We will find that $\mathcal{CS}$ is ill-suited as a combinatorial optimization technique due to its fast, finite-time convergence to homogeneous, sub-optimal populations. $\mathcal{MS}$, on the other hand, exhibits performance comparable to that of $\mathcal{CMS}$. Thus we will show that with respect to the *quality of solution* found subject to similar resource bounds,

$$\mathcal{CS} \prec \mathcal{MS} \simeq \mathcal{CMS},$$

concurring with the results obtained for MAX-CLIQUE [17]. In other words, for the 3SAT benchmark instances considered, little power is gained when going from $\mathcal{MS}$ to $\mathcal{CMS}$.

2.2 Genetic Search and Simulated Annealing

$\mathcal{CMS}$, $\mathcal{MS}$, and $\mathcal{CS}$ are exactly describable as time-homogeneous Markov chains over the space of multisets of size N. Under typical conditions, the chains for $\mathcal{CMS}$ and $\mathcal{MS}$ are irreducible and aperiodic, hence ergodic, and there exists a unique invariant probability distribution. This is not the case for $\mathcal{CS}$ due its absorbing states. Although $\mathcal{CMS}$ and $\mathcal{MS}$ are both Markov

Clause Size	$\mathcal{MS}$	$\mathcal{CMS}$	SA
200	200 (1069 gen, 5.0 min)	200 (381 gen, 2.1 min)	200 (301 gen, 3.6 sec)
430	424 (9896 gen, 1.5 hrs)	429 (7852 gen, 1.2 hrs)	430 (1763 gen, 4.5 sec)
600	587 (2761 gen, 34.9 min)	590 (4453 gen, 55.5 min)	590 (8011 gen, 5.0 sec)

Table 1: Performance results for 100-variable instances. (See text for description.)

Clause Size	$\mathcal{MS}$	$\mathcal{CMS}$	SA
800	799 (8001 gen, 2.7 hrs)	800 (2591 gen, 54.4 min)	800 (5015 gen, 4.5 sec)
2550	2472 (9921 gen, 9.1 hrs)	2500 (9919 gen, 9.3 hrs)	2534 (8041 gen, 6.0 sec)
4000	3805 (8674 gen, 20.7 hrs)	3873 (9862 gen, 24.3 hrs)	3904 (8393 gen, 8.5 sec)

Table 2: Performance results for 600-variable instances.

chains, there is one significant qualitative difference. $\mathcal{CMS}$, via its cross-over operator, allows for *non-local* transitions between states in the Hamming distance sense whereas for $\mathcal{MS}$, the transitions are *local*. We have to be careful in how we define the "distance" between finite populations. The following suffices for our purposes:

$$D(A,B) \;\equiv\; \max_{x\in B}\,\min_{y\in A}\, d(x,y)$$

where A and B are populations over binary strings of length n, and d is the Hamming distance. First, note that D is not a metric (it is not symmetric). It is easy to see that

$$D(A,\mathcal{MS}(A)) \leq i$$

when mutations are restricted to be of the i-bit kind (e.g., $i=1$), and $\mathcal{MS}(A)$ is the population obtained after applying $\mathcal{MS}$ to A. This is not the case for $\mathcal{CMS}$. Depending on A, the "distance" between A and its next state $\mathcal{CMS}(A)$ can be large. The reason that D is not symmetric, in practice, is due do the selection operator which tends to discard low-fitness elements. In the case of $\mathcal{CMS}$, the asymmetry of D is further pronounced by cross-over.

Choose some definition for the *diversity* of a population $div(A)$ with range $[0,1]$ where $div(A)=0$ if, and only if, A consists of N copies of a single element, and $div(A)\approx 1$ for a randomly chosen population. If the diversity of a population A is "low," then for many $x,y\in A$, $d(x,y)$ will be small, and we have $\max\{d(z,x),d(z,y)\}\leq d(x,y)$, $z=x\otimes y$, where "$\otimes$" denotes cross-over (e.g., two-point or uniform). Thus for low diversity populations, $D(A,\mathcal{CMS}(A))$ will be small, and $\mathcal{CMS}$ degenerates to local search. This implies that $\mathcal{CMS}$, in order to exhibit an advantage over $\mathcal{MS}$, must exploit its cross-over operator when $div(A)$ is large.

The Metropolis process (MP) is also a time-homogeneous Markov chain where the next state is achieved by local perturbation. Its state space is the set of singleton populations (i.e., $N=1$), and neighboring states with higher relative fitness values (lower energies) are preferred. Ignoring secondary considerations, $\mathcal{MS}$ can be viewed as a Metropolis process in higher dimensional space. Like MP, $\mathcal{MS}$ is restricted to *local* search, selection exerts positive bias toward high fitness states, and with small probability it moves to lower fitness states. Mutation rate takes on a role analogous to *temperature*. Clearly, $N>1$ for $\mathcal{MS}$ to avoid being a uniform random walk.

The cost of computing a single generation (or iteration) can be summarized as follows:

$$\mathcal{CMS} \;\preceq\; \mathcal{MS} \;\prec\; \text{MP} \;\simeq\; \text{SA}.$$

SA denotes simulated annealing, and "X $\prec$ Y" should be interpreted to mean that Y is *faster* than X (i.e., the cost of computing the next-state in X is greater than that of Y). In both MAX-3SAT and MAX-CLIQUE, the cost difference between $\mathcal{CMS}$ and SA is significant and straightforward to establish[2].

The previous development leads to the following observation. Given a problem domain such as MAX-3SAT, if genetic search ($\mathcal{CMS}$) is unable to make effective use of cross-over in the high-diversity regime, it degenerates into an inefficient Metropolis process. Our first goal will be to show that for large and difficult 3SAT instances, cross-over fails to be effective (i.e., $\mathcal{CMS}\simeq\mathcal{MS}$), and $\mathcal{CMS}$ has no recourse but to search as a Metropolis process. We will see that if small populations are employed, emulation can be carried out without prohibitive overhead. For easy instances, independent of problem size, cross-over does contribute to search, combining subsolutions into improved solutions thereby decreasing the generation count expended to reach an optimal solution. Since easy instances are also amenable to randomized local search,

[2]Two factors contribute to this difference, one being the overhead associated with maintaining a population, and the other being the higher cost of evaluating the fitness function which due to the non-local effects of cross-over is much costlier than if only a single bit is perturbed. In the case of MAX-3SAT, it is $O(m)$ versus $O(1)$—a clever algorithm exists—and in the case of MAX-CLIQUE it is $O(n^2)$ versus $O(n)$.

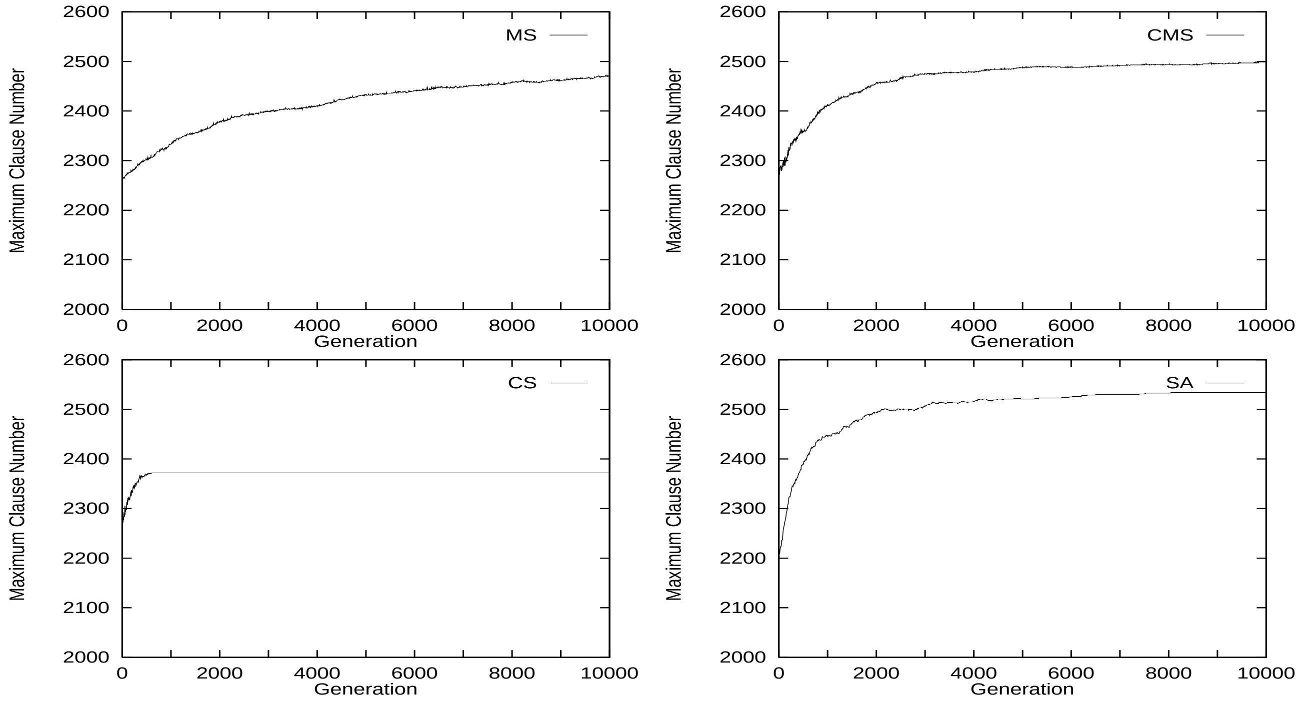

Figure 1: Evolution diagrams for $n = 600$, $m = 2550$ problem instance.

this reflects an inherent property of the problem instance itself and not the prowess of the optimization procedure employed. Small problem instances (e.g., 100-variable 3SAT formulae and 128-node MAX-CLIQUE graphs), independent of their difficulty, are solvable by genetic search within "reasonable" resource bounds. For hard instances, genetic search scales poorly as problem size is increased, indicating that small problems are unsuitable as benchmark instances due to their atypicality.

3 Performance Results

3.1 Benchmark Measurements

The benchmark instances were generated using Bart Selman's instance generator `mwff` available at DIMACS. Two problem sizes were tested, the first with 100 variables, and the second with $n = 600$. For $n = 100$, three different clause sizes were employed, $m = 200, 430$, and 600. As described in [14], for smaller problem sizes the "difficult" instances are found to lie at $\rho = m/n \approx 4.3$, not 4.25. For $n = 600$, the clause sizes tested were $m = 800, 2550$, and 4000. All instances were generated without forcing the formulae to be satisfiable. For each (n, m)-combination, three instances using different random seeds were generated. The first instance in each class is shown here. The experiments were run on a set of Sparc workstations and CPU-time measurements depict *user mode* times. A Boolean assignment to n variables was encoded in the straightforward way as a n-bit string x with $x_i = 1$ if, and only if, the ith variable was set to true. As noted in [6], SAT is one of the problem domains having a "natural" GA representation. They go so far as to suggest that other NP-complete problems should be reduced to it because of this feature. One of the motivations for selecting MAX-SAT (and MAX-CLIQUE in previous work) lies in reducing the potential for the problem encoding scheme to impede the building-block hypothesis, an important issue for TSP and scheduling problems.

Table 1 shows the maximum clause number found over 10000 generations for $n = 100$. The first number in parenthesis depicts the generation at which the maximum was first found. The last number shows the CPU-time expended until that time. The population size was set at $N = 300$. The best run over three attempts with different random seeds is shown. Two cross-over schemes, two-point and uniform, were implemented exhibiting similar performance. The results for the two-point scheme are shown here. The simulated annealing algorithm was a garden-variety implementation incorporating a linear annealing schedule.

Ignoring CPU-time differences, table 1 shows that $\mathcal{CMS}$ is able to find solutions comparable in quality to simulated annealing. $\mathcal{MS}$ performs a little worse due to the active facilitation of cross-over via diversity maintenance which has the side-effect of slowing down the convergence rate of $\mathcal{MS}$ relative to $\mathcal{CMS}$. In other settings, both $\mathcal{MS}$ and $\mathcal{CMS}$ can be made

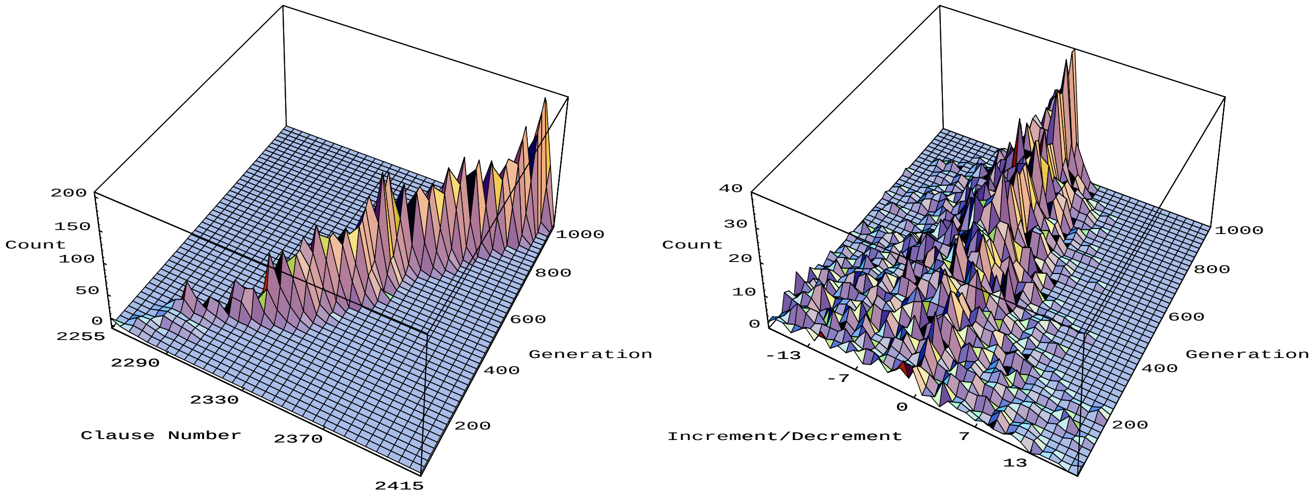

Figure 2: Performance diagram of $\mathcal{CMS}$ and its internal working. (Refer to text for description.)

to find the optimal solutions 200, 430, and 590 for the three clause sizes. The performance results indicate that for *small* problem instances, independent of their difficulty, genetic algorithms are able to find optimal solutions within "reasonable" time. This concurs with the scalability observations for MAX-CLIQUE [4, 17].

Table 2 shows the results for $n = 600$ with population size $N = 300$. Despite the increase in problem size, the $m = 800$ instance remains easy as predicted by the small $\rho = 1.33 < 4.25$ value. For the $m = 2250$, 4000 instances, there is a noticeable gap in the solution found between SA and $\mathcal{CMS}$, as well as between $\mathcal{CMS}$ and $\mathcal{MS}$. As noted in section 2.2, $\mathcal{CMS}$, to be effective, must utilize cross-over in the high-diversity regime. The runs in tables 1 and 2 were performed with this in mind, applying diversity maintenance methods such as *probability distribution damping* which throttles the bias toward high-fitness elements. The effect of increasing the population size is shown in the next section.

If attempts at maintaining diversity are aborted altogether, letting the "natural" decline as determined by N, p_m, p_c run its course, $\mathcal{CMS}$ and $\mathcal{MS}$ converge to their low-diversity regimes rapidly, subsequently searching as a Metropolis process. The solutions found are optimal for $n = 100$, and comparable to simulated annealing for $n = 600$ if sufficient CPU-time is allotted to achieve a large number of generations. The CPU-times in table 2 indicate that for the large and difficult 3SAT benchmark instances, running $\mathcal{CMS}$ and $\mathcal{MS}$ at population size $N = 300$ is, at best, inefficient relative to simulated annealing.

Figure 1 shows the time evolution of the $n = 600$, $m = 2550$ runs, inclusive the performance of $\mathcal{CS}$. The latter converges to a homogeneous, suboptimal solution (its typical behavior in all runs), and will be excluded from subsequent discussion. In the $\mathcal{MS}$ run, the slow-down in convergence relative to $\mathcal{CMS}$ due to active diversity maintenance can be discerned. Deactivating diversity maintenance alleviates this problem immediately, and $\mathcal{MS}$ achieves solutions comparable to those of $\mathcal{CMS}$ in the 10000 generation range. Figure 2 shows the *internal details* of a $\mathcal{CMS}$ run for $n = 600$, $m = 2550$, $N = 300$ over the first 1000 generations. Figure 2 (left) depicts the histogram of fitness values assigned to the 300 population elements changing over time. The peak of the distribution shifts toward improved solutions as time progresses, becoming more concentrated, indicating a decrease in diversity. Figure 2 (right) shows the search effect of cross-over at $p_c = 0.5$, and to a lesser extent also the effect of mutation at $p_m = 0.05$, during the same run. The *increment/decrement* axis represents the change in fitness value of a population element updated via cross-over. The *count* axis is the histogram of increment/decrement values over the whole population. Initially a wide range of fitness-change fluctuations are observed, its width decreasing with time as cross-over takes on a less pronounced effect, eventually degenerating to local search as diversity decreases at higher generations.

3.2 Cross-over Effects and Population Size

Figure 3 (a) shows the effect of varying the cross-over rate for the $n = 600$, $m = 2550$ instance with $N = 300$, $p_m = 0.05$, and distribution damping applied. At the highest rate $p_c = 1.0$, a solution with fitness value 2512 is found, but this is less than the 2530's achievable by both $\mathcal{CMS}$ and $\mathcal{MS}$ without distribution damping within the same time bounds. Hence, the small improvement in the solution with non-zero p_c is insignificant as a measure of cross-over's success when compared to what $\mathcal{CMS}$ can achieve operating in a

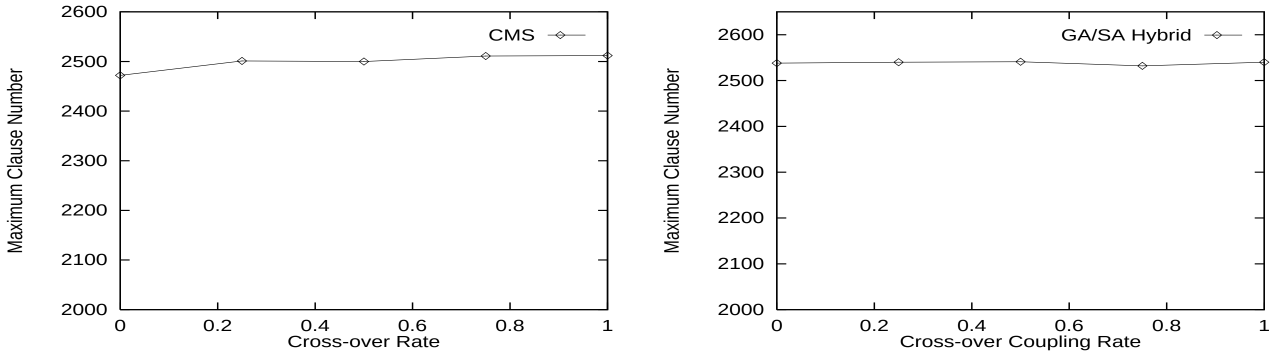

Figure 3: (a) Effect of cross-over rate on the quality of solution found. (b) Effect of cross-over coupling on the quality of solution found in GA/SA hybrid algorithm.

more greedy mode. Similar results hold for $p_m = 0.1$ and 0.01. At large values of p_m, the increased variance of the $\mathcal{CMS}$ process renders it ineffective below the 10000 generation range.

Figure 4 shows the effect of increasing the population size for the $n = 600$, $m = 2550$ problem instance when distribution damping is active. Figure 4 (left) shows the improvement in solution found as N is increased from 50 to 1200. The number of generations was fixed at 10000. Figure 4 (right) shows the corresponding CPU-time expenditure which approaches 90 hours for $N = 1200$. The solution found by $\mathcal{CMS}$ at $N = 1200$ is 2525. With population size $N = 5000$, $\mathcal{CMS}$ finds a solution of 2523 at generation 2000, requiring nearly 40 hours to do so. The gap in $\mathcal{CMS}$ and $\mathcal{MS}$ performance is due to diversity maintenance and can be removed by simply deactivating it. The results show that $\mathcal{CMS}$ performance is not enhanced by extraneous mechanisms introduced to facilitate cross-over's success rate. Increasing the population size slows down convergence in a *natural* way, improving the solution found. It does not, however, occur at sufficiently early generations to warrant its high computational cost given the small gain in the solution found. This concurs with the observation of slow increase in the quality of solution found as a function of population size in the case of challenging MAX-CLIQUE instances [4], where population sizes up to 32000 were tested using a CM-5.

3.3 Emulating Simulated Annealing

Figure 5 shows the result of running $\mathcal{CMS}$ and $\mathcal{MS}$ without any diversity maintenance mechanism at population size $N = 20$ for up to 50000 generations. The mutation and cross-over rates were set at $p_m = 0.05$ and $p_c = 0.5$, respectively. As the graphs indicate, the qualitative behavior of $\mathcal{CMS}$, $\mathcal{MS}$, and simulated annealing are very similar. In this particular run, $\mathcal{CMS}$ finds an assignment with fitness value 2527, $\mathcal{MS}$ finds 2533, and SA finds a solution of 2541. The costs for $\mathcal{CMS}$ and $\mathcal{MS}$ are in the 3 hour range whereas SA expends 27 seconds. By the tweaking of parameters, further improvements in performance can be achieved. The diversity plot for the $\mathcal{CMS}$ run shows a sharp decrease in population diversity occurring in just 30 generations, remaining near 0 thereafter. If one is "oblivious" to the internal workings of the $\mathcal{CMS}$ process and its computational cost relative to simulated annealing, then for small population sizes, one may observe $\mathcal{CMS}$ equalling, and sometimes outperforming, SA. That is, genetic search can be made to emulate simulated annealing with "reasonable" overhead.

3.4 Cross-over Effects in Hybrid Environment

It is possible to adopt a broader interpretation of cross-over as reflecting the "two good solutions can be combined into a better one" thesis. In this vein, we considered a set-up where N independent simulated annealing processes were run, each selecting a mate to cross-over with at some fixed rate as determined by a coupling parameter. The resulting candidate solution was subject to the standard SA acceptance function. Similar hybrid algorithms have been considered in [3, 10]. If we let cross-over rate p_c denote the coupling strength, then $p_c = 0$ reduces the system to running N independent copies of SA with different initial states.

Figure 3 (b) shows the effect of the coupling rate on the performance of a $N = 100$ system over 10000 iterations for p_c values ranging from 0 to 1. The curve is flat, indicating that candidate solution generation based on the cross-over principle does not enhance the search process.

4 Discussion

We have presented an investigation of genetic search and its effectiveness at solving the MAX-SAT problem. The performance results indicate that for large and difficult 3SAT instances, cross-over fails to be effec-

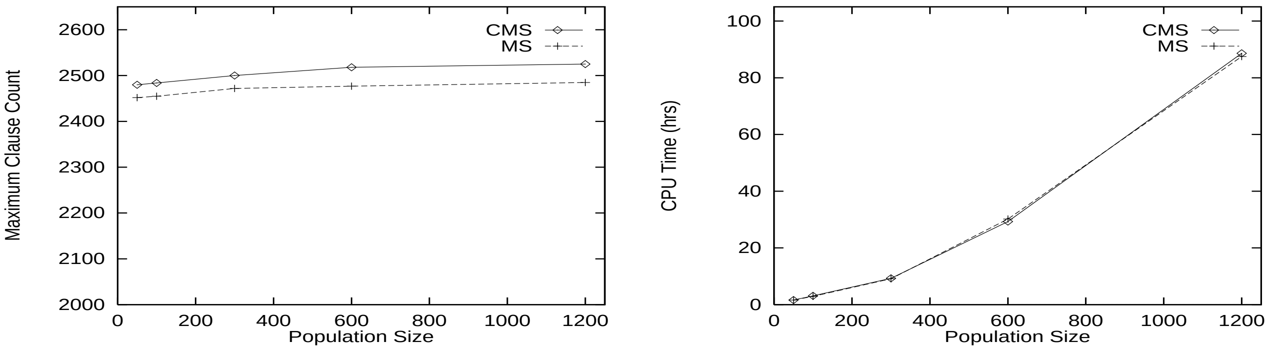

Figure 4: Effect of population size on the quality of solution found and its cost.

tive, and genetic search is consistently outperformed by simulated annealing. For small problem sizes independent of difficulty, or for simple instances independent of problem size, genetic search is able to find optimal solutions but at a computational cost higher than simulated annealing.

The decomposition method allowed us to evaluate cross-over's contribution by comparing the performance of $\mathcal{CMS}$ and $\mathcal{CS}$ against $\mathcal{MS}$. The correspondence of the latter to the Metropolis process provided a bridge by which a qualitative comparison between $\mathcal{CMS}$ and simulated annealing could be established. We have also shown the aptitude of genetic search at emulating simulated annealing, the computational overhead of which can be reduced by employing small populations.

It is inherently more difficult to demonstrate negative than positive results experimentally. One can never claim to have exhausted all possible avenues of attack, and this paper makes no claims in this regard.

The MAX-SAT results in conjunction with previous work on the MAX-CLIQUE problem [4, 17] suggest that for large and difficult problem instances, the building-block hypothesis may be an inadequate characterization, rendering cross-over's role marginal given its high computational cost. For linear and "near-linear" problems, it is possible to show that genetic algorithms are effective at finding optimal solutions [16, 18]. In fact, it can be argued that linear and near-linear problems are, in some sense, the best representant problem class for the building-block hypothesis. It is for these problems where schemas have meaningful "worth" independent of the surrounding context, and combining two good subsolutions has a nonnegligible probability of resulting in an improved solution.

Our experimental results are of too limited scope to be extrapolated in general. However, as a focus point, we advance the conjecture that any NP problem efficiently solvable by genetic search is also efficiently solvable by randomized local search, in particular, simulated annealing. This problem seems very challenging, and resolving the conjecture in the positive or negative may be rewarding, if perhaps not equally so.

Acknowledgements

The author was supported in part by NSF grant CCR-9204284. Many thanks to Bob Carter and an anonymous referee for detailed and informative comments. Due to space restrictions, the reference section has been shortened. The author can be contacted at `park@cs.bu.edu`.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 14–23, 1992.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 2–13, 1992.

[3] T. Boseniuk, W. Ebeling, and A. Engel. Boltzmann and Darwin strategies in complex optimization. *Physics Letters A*, 125(6/7):307–310, 1987.

[4] B. Carter and K. Park. Scalability problems of genetic search. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, pages 1591–1596, October, 1994.

[5] L. Davis (ed.). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[6] K. De Jong and W. Spears. Using genetic algorithms to solve NP-complete problems. In *Proc. 3rd International Conf. on Genetic Algorithms*, pages 124–132, 1989.

[7] D. Fogel and J. Atmar. Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems. *Biological Cybernetics*, 63:111–114, 1990.

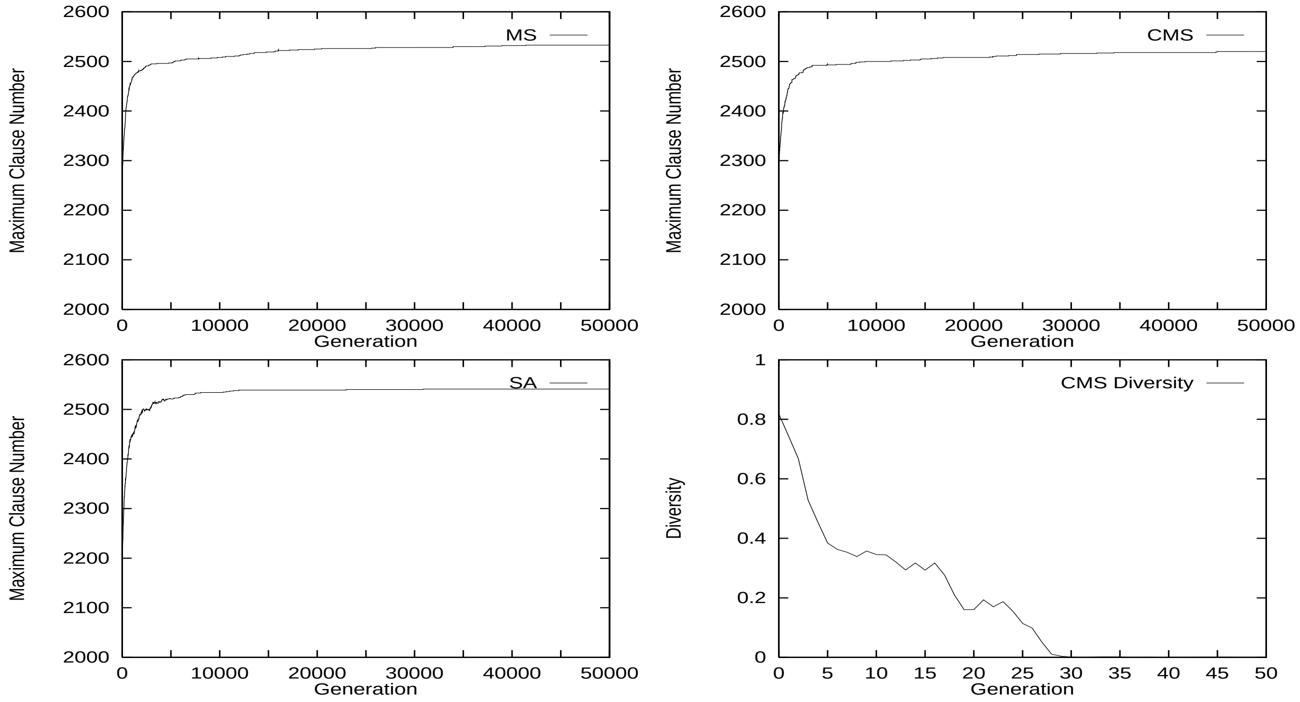

Figure 5: Evolution diagram for $n = 600$, $m = 2550$ instance with small population.

[8] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. *Preprint*, 1992. (To appear in *Machine Learning*.)

[9] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[10] D. Goldberg. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, 4:445–460, 1990.

[11] J. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[12] P. Jog, J. Suh, and D. Gucht. Parallel genetic algorithms applied to the traveling salesman problem. *SIAM Journal on Optimization*, 1(4):515–529, 1991.

[13] D. Levine. A genetic algorithm for the set partitioning problem. In *Proc. 5th International Conf. on Genetic Algorithms*, pages 481–487, 1993.

[14] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions for SAT problems. In *Proc. AAAI-92*, pages 459–465, 1992.

[15] H. Mühlenbein, M. Gorges-Schleuter and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[16] K. Park. A lower-bound result on the power of genetic algorithms. In *Proc. 5th International Conf. on Genetic Algorithms*, pages 651, 1993.

[17] K. Park and B. Carter. On the effectiveness of genetic search in combinatorial optimization. In *Proc. 10th ACM Symposium on Applied Computing, Genetic Algorithms and Optimization Track*, pages 329–336, 1995.

[18] Y. Rabinovich and A. Wigderson. Analysis of a simple genetic algorithm. In *Proc. 4th International Conf. on Genetic Algorithms*, pages 215–221, 1991.

[19] D. Schaffer, R. Caruana, and L. Eshelman. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proc. 3th International Conf. on Genetic Algorithms*, pages 51–60, 1989.

[20] D. Schaffer and L. Eshelman. On crossover as an evolutionary viable strategy. In *Proc. 4th International Conf. on Genetic Algorithms*, pages 61–68, 1991.

[21] R. Unger and J. Moult. Genetic algorithms for 3D protein folding simulations. In *Proc. 5th International Conf. on Genetic Algorithms*, pages 581–588, 1993.

[22] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.