A New Approach for Heterogeneous Hybridization of Constraint Satisfaction Search Algorithms

Debasis Mitra and Hyoung-rae Kim
Department of Computer Science, Florida Tech. FL.
dmitra@cs.fit.edu, hokim@fit.edu

Abstract

In the recent years, CSP's have come to be seen as the core problem in many applications. We propose here a hybrid algorithm (MC-FC) that combines two different search methods, where a problem is solved partially with a random answer using Min-conflict algorithm (MC) and then the partial solution guides the systematic search algorithm (Forward-checking, FC) to solve the remainder of the problem. The specific problems used for this study are primarily the graph coloring problems with various degree of connectivity. The time complexity of the MC-FC is much better than the pure MC and FC under many circumstances. Furthermore, its time-complexity is more predictable over multiple experiments compared to that of the pure MC or FC, and the former algorithm uses less storage than the pure algorithms. We attempt to motivate our heterogeneous hybridization technique with some preliminary experiments and simple manual calculations.

1. Introduction

Algorithms for solving CSP's usually fall into one of the three main categories: systematic search algorithms, local search algorithms, and hybrid algorithms. Systematic algorithms start from a state where all variables are empty and are subsequently instantiated until the state of a complete assignment. The disadvantages of these algorithms are thrashing, redundant work, and late detection of the conflict - starting with a wrong variable will explore the whole sub trees [9]. Local search algorithms (e.g., min-conflict [10], GSAT [15], tabu search [5], etc.) perform an exploration of the search space by selecting the best possible assignments locally. These algorithms start with a full assignment to all variables, and then repair inconsistent assignments. The disadvantage is that there is still a danger of falling into local minima. These methods are also incomplete – if a problem has no solution it will not terminate. Hybrid algorithms, on the other hand, are integration between local and systematic searches. The hybrid approaches have led to good results on large-sized problems. There have been many works combining local and systematic search algorithms by coordinating between the two [8,11,14,12,13,16,19].

In order to enhance the systematic search of Forward-checking algorithm (FC) we start from a randomly built partial solution. Then the FC algorithm may succeed in finding a solution or fail, more efficiently since it could prune many inconsistent domain values from the beginning. Our approach is to obtain a partial random solution for the first portion (random part) of the problem (Z_K, D, C, where Z_K is a k-subset of the set of variable Z, D a set of domain of values for each variable, and C a set of constraints between the variables [18]) in an arbitrary (or any natural) ordering of the variable 1 through N. Then our approach uses that partial solution to guide the systematic search for the remainder (systematic part) of the problem (Z_{N-K}, D, C). This guidance reduces the workload for the systematic search (especially for the Forward Checking algorithm) by pruning the initial search space because every complete solution of the problem contains a partial solution as a subset (although every partial solution may not lead to a complete solution).

We have used Min-conflict algorithm (MC) [10] for the random part and Forward-checking algorithm (FC) [6,18] for systematic part. We call this technique a *heterogeneous hybridization* scheme (for the obvious reason). A "homogeneous" hybrid algorithm has a tighter integration, e.g., a systematic search may act for propagating constraints or checking validity of assignments at each step of repair of a local search [14,19]. Our experiments primarily focus on the graph coloring problems, but we have also worked with the N-queens problem.

Our primary contribution is the new heterogeneous way of hybridizing two completely separate parts, and consequent method of experimenting with problem structure. The advantages of this approach are to reduce the time complexity and also to yield more *stable* complexity (by *stable* we mean having a small *confidence interval* for the mean over multiple experiments), compared with what the pure local or the pure systematic search algorithm achieve. The disadvantage of, the algorithm is that not complete, i.e., it may not find a solution even if one exists. To compensate for this disadvantage we have hinted two other variations of our algorithm that are complete. Our objective is to propose the heterogeneous way of hybridizing algorithms, rather than finding the fastest algorithm for the graph coloring problem or the N-queens problem. For this reason we have experimented with two pure versions of algorithms, namely the MC and the FC, rather than their more efficient versions.

The rest of this paper is organized as follows: Section 2 describes partial solution and the motivation of heterogeneous algorithm. Section 3 details MC-FC algorithm. Section 4 describes our experiments. Section 5 analyzes the results from the experiments. Section 6 discusses related work in hybrid algorithms. Section 7 summarizes our findings and suggests possible future work.

2. Partial Solutions

When the number of nodes is large, FC does not remove inconsistent domain values efficiently in general. In order to alleviate this problem we start with a randomly generated partial solution. Then FC may work faster since many inconsistent domain values could be pruned initially. Our algorithm divides a problem into two partitions ($Z_K \cup Z_{N-K}$, D, C), for N variables and 1<=K<=N, and we run the two different types of search algorithms over the two different partitions. For example, in 16-queens problem, if K is 2, the initial problem is to put two queens in a 16 by 2 board. Since we are using a random partial solution for guiding the subsequent search over the complementary partial problem, studying the characteristics of partial solution is important. Below we study the complexity (and another relevant parameter) of the partial problem solving process by the two search techniques.

Firstly, as mentioned before, the Informed Backtracking algorithm (MC) with Min-Conflict heuristics [10] is chosen for the random part. The MC starts with random initial values and is guided by the Min-conflict heuristic, which tries to find the best alternative in its search-path that conflicts minimally. Since we plan to study the change of time complexity against different partial problem sizes (we use variable K as the partitioning parameter, 0<=K<=N, where N is the variable number in the CSP), this algorithm is the appropriate choice, as we wanted to find a partial solution quickly. We have varied K from 1 through N in this study. Las Vegas algorithm [3] also depends on randomness, and also is a potential candidate. However, it is slower than the MC algorithm [6] in general.

Secondly, we also need to measure the change of complexity for the systematic search part (K+1 through N variables) by varying K (from 1 through N). Kondrak and Beek [9] compared the efficiency of several systematic search algorithms theoretically and empirically. Even though there are faster algorithms than FC (e.g., FC_CBJ) [9], the FC algorithm, being a "pure" algorithm, is more suitable and logistically easy for measuring how well the random initial value affects the final results. This is why we have picked up the FC algorithm for the systematic part.

Both the parts above are over complementary partitions of the problem space. Thus, for an arbitrary ordering of variables, one part is 0 through K, and the other part is K+1 through N, where K=0 indicates pure FC, and K=N indicates pure MC.

One of the ways to motivate such a heterogeneous hybridization is to experiment separately over each partition independently (without using the second algorithm), and then to aggregate the result from the two cases. If the first part (running MC) sharply increases and the second part (running FC) decreases as K increases, then we expect the hybridization to produce better results. In sections 5.2 and 5.3 we presents results from such motivating experiments. We control two parameters: K (size of the partial solution), and R (number of allowed repetitions with random starts for MC). We can assign a large value to R in order to increase the *accuracy*. We measure *accuracy* by the number of successes in finding a solution over multiple experiments – the probability of success.

3. The MC-FC Algorithm

In this section we explain the hybrid MC-FC algorithm. We solve the partial problem using MC from 1 through K (1<=K<=N), and then use FC for the remainder of the problem. Partial solutions are fed to FC and the FC starts with pruning of the remainder of the variables' domains accordingly. However, MC may feed a partial solution that does not lead to a complete solution for the whole of the problem. If the FC does not find a solution, the process is repeated, up to the pre-assigned maximum repetition number (R) or until FC finds a solution. The value of K is varied not only to prove our conjecture that hybridization is more efficient than each of its component algorithms, but also to find the optimum value of K for the best time-complexity.

```
procedure MC-FC (k)
  Count = 0
  While Count < R do
    Count++;
    Initialize Z, D, C;
    COMPOUND_LABEL = MC(k, Z, D, C);
    Result = FC(k, COMPOUND_LABEL, Z, D, C)
    if Result == valid then
      return result;
    else return false;
    end if
  end while
end procedure
```

Figure 1. MC-FC algorithm

This algorithm in Figure 1 is not complete. A modified complete algorithm (MC-FC-2) can be designed to alleviate this weakness. As an enhancement we add a hash table to keep all the tested COMPOUND_LABELs (partial solutions) over different repetitions (<=R). A new COMPOUND_LABEL is checked against this no-good database before it is fed to the FC. This would work when the size of the possible COMPOUND_LABEL is small. However, over a very large number of repetitions, the hash table size also increases: in the worst case the space complexity of the hash table becomes $O(|D|^{|Z|})$, with |Z| number of variables and |D| values in each variable's domain. Another alternative algorithm (MC-FC-3) will be to revise MC to actually backtrack when FC fails rather than to start from the scratch in the latter case, i.e., let MC to keep track of what it has visited, which is similar to the work by Zhang [20]. This is also a complete algorithm and the asymptotic time complexity is the same as the original MC-FC. The disadvantage of MC-FC-3 is that the solution could depend on the initial assignment too much. Thus, it loses the power of randomness and is likely to backtrack more than necessary. As one of the purposes of this work is to prove the new paradigm of hybridization, we use the original MC-FC that yields the most randomized partial solution.

4. Experiments

Data. We used N-node graph coloring problems over various *percentages of connected nodes* (CN), and the 16-queens problem. Graph coloring problem is a well-known NP-complete problem [7]. We have chosen N=16, 18, and 20 for the reported set of experiments. To create a graph, we connected N nodes randomly depending on the pre-assigned value of CN. We created various N-node graph coloring problems over different graph connectivity CN=0.55, 0.7, and 0.8, where CN = ((number of arcs) / (N(N-1)/2)), for N nodes, and with different number of colors (|D|). We use the minimum number of colors needed for different values of CN (determined by separate preliminary experimentation). We have used various CN' s representing different problem structures for the graph coloring problems in order to cover various types of problems.

Criteria. We have analyzed the number of consistency checks (CC) for the time complexity, and statistical confidence interval (CI) over multiple experiments for the *stability*. The most common methods to measure the time complexity are visited nodes (VNs) and consistency checks (CC) [9]. Since CC subsumes VN, we mainly use CC in our experiments. We define a data point (mean) to be *stable* when the confidence interval [1] is small on the mean values of a distribution of CC over repeated experiments (up to R or until a solution is found). In other words, the CI indicates the quality of the data points.

Procedure. We have run FC and MC for experimenting over the partial problems (motivating experiments), and the hybrid MC-FC algorithm over the whole problem. Implementations used Java™ and our experiments are performed on a Sun Ultra 60™ workstation. We have averaged our results around 300 different runs of the three programs. There are three different set of experiments: for the random part (MC), the systematic part (FC), and the hybridized algorithm over the whole problem.

5. Analysis

5.1. The Complexity of MC Over Partial Problems

Procedure. We measured the growth rate of CC of the MC algorithm against the size (K) of the partial problem. For example in the 16-queens problem, if K is 2, the problem is to put two queens in a 16 by 2 board. As K grows up to 16, the problem becomes the complete 16-queens problem. So, such a partial problem is really an *N×K queens* problem.

Results. The results are shown in Table 1. Nodes mean the number of nodes in N-node graph coloring problem. CN stands for average connected nodes. C stands for the minimum number of colors to solve the problem. The results show that the CC increases sharply toward the high K value. The CC for N=16, for CN= 0.7 or 0.8, and with K=16, is abnormally higher than the CC value for the CN=0.55 values, which means that those points are more difficult for MC to solve than the other points (by more difficult we mean it takes more time). We did not show CN=0.3 and 0.9 because those have relatively low CC values. 16-queens problem also yield similar results. As K approaches N, CC increases sharply. We have actually observed this pattern over many other N values for the N-queens problem. We did not show any CI value because the average of CC alone explains the changes well here.

Table 1. **CC ($x10^3$) with N-graph coloring problem over various CN in MC part**

K	16-queen	16 graph coloring				18 graph coloring			20 graph coloring		
		CN	0.55	0.7	0.8	0.55	0.7	0.8	0.55	0.7	0.8
		C	5	6	8	5	8	8	6	8	10
1	0		0	0	0	0	0	0	0	0	0
2	0		0	0	0	0	0	0	0	0	0
3	0		0	0	0	0	0	0	0	0	0
4	0		0	0	0	0	0	0	0	0	0
5	0.1		0.1	0.1	0.1	0	0.1	0	0	0	0
6	0.2		0.1	0.1	0.1	0.1	0.1	0.1	0	0.1	0.1
7	0.3		0.1	0.1	0.2	0.1	0.2	0.2	0.1	0.1	0.3
8	0.4		0.2	0.2	0.3	0.2	0.2	0.3	0.2	0.2	0.3
9	0.6		0.3	0.3	0.3	0.3	0.4	0.4	0.3	0.4	0.4
10	0.7		0.4	0.4	0.5	0.4	0.4	0.5	0.5	0.5	0.7
11	0.9		0.5	0.5	0.7	0.5	0.6	0.6	0.6	0.6	0.8
12	1.2		2.2	0.7	0.8	0.6	0.8	0.8	0.8	0.8	1.0
13	1.5		2.3	0.8	1.0	0.7	1.0	1.1	0.8	0.9	1.2
14	2.0		2.5	1.1	2.6	0.9	1.2	1.2	1.1	1.2	1.4
15	3.0		2.3	2.4	2.1	5.3	1.4	1.4	1.2	1.5	1.7
16	7.1		4.4	3357.9	5643.1	190.1	10.4	100.6	2.4	2.2	1.8
17						293.5	6.9	120.6	6.3	2.8	19.0
18						181.4	9.2	149.6	2.7	7.0	33.5
19									4.3	1088.8	56.2
20									1642.4	1184.9	176.3

5.2. The Complexity in the Systematic Part (FC) Over the Partial Problems

Procedure. Analysis of the systematic search part is slightly different from the previous mode of study for the random part. We have randomly generated partially satisfying assignments for variables numbered 0 through K, and then used that partial solution to initialize the FC algorithm for running over the rest of the problem (for variables K+1 through N). If the FC fails to find a complete solution of the whole problem (1 through N), then we reinitialize again in a similar way as before and rerun FC, i.e., we repeat the experiment. By repeating the experiment we derive a statistical average of CC for the FC algorithm, for different values of K. An upper limit R for the number of such repetitions is pre-assigned (which is also varied). We have studied *accuracy* against K in order to make sure we use high value of R in our subsequent experiments with FC algorithm for investigating only the "solvable" problems. The max repetition number R was set to 100,000.

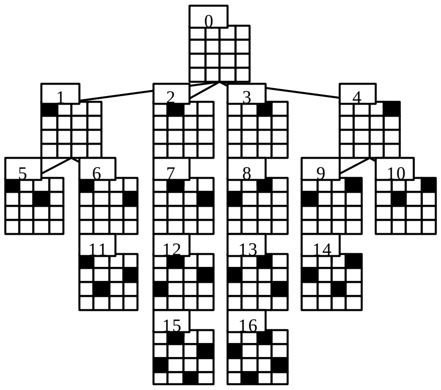

Figure 2. 4-queens problem

We explain how we calculate the time complexity and the *accuracy* for the systematic part. We use 4-queens problem as an example since its domain size is small, and we only count VNs (visited nodes) to simplify the counting process (although we have used CC for the experiments). All visited nodes are expanded by an FC algorithm as shown in the Figure 2. K is the number of variables that are assigned to a value initially as a partial solution tuple, with a varying K. If the attempt (FC) fails to find a solution we rerun it one more time, so, R=2 here. A row indicates the corresponding variable, and the columns indicate the domain values of any variable.

For K=1 there are four possible starting assignments for the first variable indicated by the nodes {(1),(2),(3),(4)} (see Figure 2). The possibility of an occurrence of the node (1) is 1/4, and the VNs is 3 and it always fails (for every random attempt of initializing at that node). The remaining occurrences (nodes 2, 3, and 4) are calculated in the same way. Every initial node causes FC to visit exactly three nodes following the respective initialization. So, the average number of VNs is 3 and the standard deviation (SD) is 0. The possibility of success (*accuracy*) is ½(50%) in average. Suppose we allow rerun (say, for pre-assigned R=2), the rerun happens only when FC fails, i.e., when initial nodes is 1 or 4. So, there are only 10 such cases [{(1),(1)}, {(1),(2)}, {(1),(3)}, {(1),(4)}, {(2)}, {(3)}, {(4), (1)}, {(4), (2)}, {(4),(3)}, {(4),(4)}]. As an example, the situation of {(1),(2)} means the first initialization was at node (1) and the second rerun started with node (2). Each case in this scenario has 1/10 possibility of occurrence; the possibility of success is 6/10 (60%); VN is 6 for 8 cases and 3 for 2 cases (5.4 in average).

For K=2, it has 6 possible starting points {nodes (5), (6), (7)… (10)}. The possibility of success is 2/6 (33%); the average VNs is 6/6 (=1). When we rerun one more time (say, R=2), it has 26 possible starting points [{(5),(5)}, {(5),(6)}, {(5),(7)} … {(10),(10)}]. The possibility of occurrence for each case is 1/26, the possibility of success is 10/26 (38%), and it has 40/26 (=1.5) VNs in average - the SD is 0.9. We can observe from this simple calculation that as we increase K to 2 and allow rerun, the possibility of success (*accuracy*) and VN are both reduced. We showed how we calculate the possibility of success, in order to provide an insight into the relationship between success and the number of visited nodes (or CC).

Results. In 16-queens problem, as K increases we can clearly see that the CC reduces. As we allow more repetition, the *accuracy* reaches to 100%, which is expected because we are solving the partial problem, starting with randomly chosen different initial solutions, and repeating this until we get the solutions. Table 2 shows that the CC drops as K increases (K represents the size of partial solutions). 16-node graph coloring problem with CN=0.7 shows the same type of changes. The one with CN=0.8 of N=16 had low CC values in the middle. We did not show CI because the average of CC alone shows the changes well enough.

Table 2. CC ($x10^3$) with N-graph coloring problem over various CN in FC part

K	16-queen	16 graph coloring				18 graph coloring			20 graph coloring		
		CN	0.55	0.7	0.8	0.55	0.7	0.8	0.55	0.7	0.8
		C	5	6	8	5	8	8	6	8	10
0	10.0		0.4	761.2	972.9	0.6	0.8	24.1	942.3	1.0	1.2
1	4.2		0.3	84.5	937.2	59.8	1.1	24.1	128.5	770.4	1.2
2	2.3		1.3	235.4	227.8	130.4	1.9	28.0	158.1	801.9	1.6
3	1.6		1.1	120.4	260.8	11.5	1.4	12.0	19.0	143.6	1.2
4	1.7		2.7	147.9	39.6	3.8	1.3	14.9	22.4	162.6	5.3
5	1.0		1.2	71.4	16.3	2.6	2.4	9.9	17.1	45.7	2.9
6	0.5		1.1	20.7	15.1	8.2	2.6	9.6	9.0	50.3	20.6
7	0.1		1.1	13.1	6.3	2.8	7.4	12.6	10.4	30.0	17.2
8	0		1.5	6.9	6.7	7.7	3.9	14.7	7.2	10.0	61.3
9	0		1.1	4.9	11.1	11.4	1.9	37.0	4.5	4.4	14.5
10	0		2.5	4.7	19.1	11.6	4.5	19.2	4.8	23.0	15.2
11	0		1.1	3.7	60.9	19.7	1.3	24.5	5.5	8.9	5.7
12	0		0.2	3.5	68.8	9.7	1.0	12.8	4.8	9.8	3.6
13	0		0.2	5.0	43.5	9.1	1.6	2.6	6.3	10.0	2.4
14	0		0.1	3.1	8.2	11.1	2.6	3.7	4.2	3.0	1.7
15	0		0.1	0.3	5.2	3.8	1.9	3.4	5.9	7.0	1.6
16	0		0	0	0	0.5	0.4	0.4	4.0	5.9	1.8
17						0.1	0.1	0.2	2.4	3.2	1.1
18						0	0	0	2.5	3.0	1.5
19									0.5	0.1	0.4

5.3. The Complexity of Hybridization Over Varying K-values

Procedure. In the following experiments with our hybrid algorithm, we allow large value of R in order to have 100% *accuracy* in each test. We also vary K from 0 through N, where K=0 corresponds to the pure FC algorithm and K=N corresponds to the pure MC algorithm.

Results. The results of the hybridized algorithm were reported in Table 3. ANOVA (single factor) clearly shows that there were differences among different K values, by yielding less than 0.05 of P value. We can also clearly observe that FC is misdirected away from the solution in the search space in the case of N=20, CN=0.55 and K=0 because its value is so high compared to the value of N=20, CN=0.7 and K=0. In nodes=20, CN=0.7 with pure FC algorithm (K=0) it had very low CC, but when we started the MC-FC algorithm with the starting position randomly (K=1) the value of CC became larger, which shows the brittleness of FC.

Case with nodes=16, CN=0.7 and 0.8 are more difficult than that with CN=0.55; with nodes=18, CN=0.55 and 0.8 is more difficult than that with CN=0.7; and with nodes=20, CN=0.55 and 0.7 are more difficult than CN=0.8. Whenever the problem is difficult, it has the minimum CC value in the middle (other than the CC for the pure FC). The results show that in 16-graph coloring problem with CN=0.7, K=8 provided the best results (CC=8385 and CI=2762). The upper bound and lower bound were 11,147 and 5,623; it overlapped only with its neighboring interval (K=7 and K=9), but not with the other data points, which means it is better than the others in confidence level of 0.05. Furthermore, the CI of K=8 (2762) was lower than that for other K values generally. Since every different case of K had the same number of sample data, the K with the lower CI is more *stable* than the ones with higher CI values.

The CI values are also low for each data point where CC values are found to be minimum. The "CI M" stands for CI values corresponding to the minimum CC values. This value can be used for checking whether they are statistically different from the other neighboring data points. We omitted the other CI values due to the space limitation.

Table 3. The results with N-graph coloring problems over various CN in MC-FC (x10^3)

K	16-queen	16 graph coloring				18 graph coloring			20 graph coloring		
		CN	0.55	0.7	0.8	0.55	0.7	0.8	0.55	0.7	0.8
		C	5	6	8	5	8	8	6	8	10
0	148.4		0.4	761.2	972.9	0.6	0.8	24.1	942.3	1.0	1.2
1	69.7		0.3	111.4	939.0	87.1	0.9	24.1	252.9	866.5	1.2
2	35.6		1.8	314.9	262.0	137.7	2.1	27.7	127.7	866.5	1.5
3	28.6		1.0	116.9	107.6	21.9	1.6	12.9	81.6	1726.5	1.3
4	22.3		2.3	136.7	30.7	4.4	1.4	15.2	15.4	105.3	2.9
5	18.3		0.8	92.1	33.2	2.0	1.5	7.3	22.2	40.6	3.0
6	11.6		1.0	31.2	14.7	9.4	2.1	20.5	9.4	41.1	33.6
7	13.2		1.0	15.5	5.4	6.4	5.2	12.6	12.8	26.8	9.7
8	18.4		1.4	8.3	8.0	13.1	3.3	10.0	8.5	13.5	103.5
9	27.0		2.9	10.0	19.0	14.2	2.6	85.4	7.9	8.1	16.5
10	41.5		5.5	14.5	36.5	22.3	4.9	31.2	6.9	24.9	23.5
11	65.1		3.6	23.6	164.0	32.5	2.8	37.1	17.3	14.4	10.1
12	79.5		2.7	14.2	215.2	28.5	3.0	31.8	12.6	20.9	6.7
13	80.5		3.6	23.3	189.2	38.2	4.0	8.2	17.1	25.6	6.8
14	67.1		2.7	30.1	106.0	55.3	17.1	13.6	14.3	10.9	5.6
15	55.4		3.0	40.4	121.5	89.7	7.9	15.0	21.5	27.6	6.6
16	46.1		2.8	3806.2	7702.4	735.3	17.6	170.0	34.6	42.5	5.5
17						220.9	7.9	173.6	32.6	61.3	52.6
18						254.8	10.5	128.8	35.3	129.7	125.5
19									32.0	3107.2	70.8
20									1471.8	1397.8	80.6
Best K	6		1	8	7	5	0	5	9	9	1
CI M	0.9		0	2.7	1.6	0.5	0	4.1	2.9	2.5	0

All of the results with the other N-queens problems (20, 24, 26, 28, 30, and 32) have similar patterns as we vary the K value, i.e., all N-queens problems had its minimum CC when K is around in the middle. 16-qeens problem has its lowest CC value at K=6.

The best K value for each case are indicated by highlighting the complexities or the CC value that is the lowest in each column of Table 3. The best K varies with the problem. However, we can observe that the best K value is around in the middle for the harder problems (e.g., N=16 and CN=0.7 is harder than N=16 and CN=0.55) in terms of computational requirements.

We have experimented also with the complete algorithm MC-FC-2 and the results are very similar to that for MC-FC. However, as we have discussed before that MC-FC-2 is not scalable for large problems. We do not present those results here for the lack of space.

6. Related Research

Jussien and Lhomme [8] presented a new hybrid technique. It performed a local search over partial assignments instead of complete assignments, and used filtering techniques and conflict-based heuristic to efficiently guide the search. They used a tabu search algorithm. Zhang [20] proposed a hybrid algorithm that used automatic symmetry breaking method as a preprocessing step, and then the propositional satisfiability procedure is called for completing the search. This method is similar to our MC-FC-3. However, the MC-FC algorithm allows randomness in the MC part by completely separating them. Schaerf [14] proposed a solution technique that combined backtrack-free constructive methods and local search techniques. This technique incrementally constructed the solution performing a local search on partial solutions each time the construction reached a dead-end. Pargas and Ludwick [11] combined genetic algorithms with search algorithms, providing clues that ideally accelerate each other's ability to find the optimal solution. Pesant and Gendreau [12] proposed a novel way of looking at local search algorithms. They concentrated on the neighborhood exploration. Shaw [16] combined local search and constraint programming. His "Large Neighborhood Search" explored a large neighborhood of the current solution by selecting a number of "related" customer visits, and re-inserting these visits using a constraint-based tree search. He used Limited Discrepancy Search during the tree search to re-insert visits. Richards and Richards [13] explored the performance of a complete non-systematic search algorithm learn-SAT, which was based on restart-repair and learning no-goods. None of the previous works has decomposed the problem space completely as we are proposing.

Hogg et al. [7] presented and experimentally evaluated (over the graph coloring problem) the hypothesis that *cooperative* parallel search is well suited for hard graph coloring problems near phase transition zone. They found that simple cooperative methods could often solve such problems faster than the same number of independent agents. Our method uses single agent unlike their method.

7. Conclusions and Future Work

We have proposed here a technique for hybridizing algorithms that is a systematic search guided by a partial solution. We have used the MC algorithm to get the partial solution for the *random part* and the FC algorithm for the *systematic part* of the problem. We have also quantified the degree of hybridization with the size (K number of initial variables in an arbitrary total order of the variables) over which the local search runs and we have experimented by varying K. The best results with the lower time-complexity and the smaller CI (confidence interval, for the distribution of that complexity over multiple experiments) are observed at some mid-range K values between 0 and N. This not only proves the power of the hybridization (as we have motivated with the preliminary experiments), but also quantifies where such optimal efficiency is expected.

We emphasize that the purpose of this article is to introduce a new way of hybridization rather than to introduce a faster algorithm. For instance, Sumitaka [17] proposed a faster algorithm that finds one solution for the N-queens problem. Our contribution is to prove the conjecture that the heterogeneous hybridization is promising. This is also why we used standard MC and FC algorithms.

Another advantage of our heterogeneous hybrid algorithm is that it uses less memory than the pure MC or FC. This is because as FC starts to execute, the expanded nodes by MC are deleted, except just the partial solution that MC returns. Also, MC-FC "forgets" the FC algorithm's work when it repeats back the MC on FC's failure in finding a solution. Therefore, it either keeps only the nodes expanded by MC (up to variable K) or those by FC (for variables K+1 through N). Our hybridization technique is also quite easy to implement because of its heterogeneous nature. Our experimentation with the degree of hybridization by varying K opens a lot of possibility for creating such heterogeneous combination between algorithms (e.g., combination of other local searches and systematic searches) and we will pursue those possibilities.

We have "invented" a ($N\times K$)-queens problem, where the number of queens is the smaller of the two numbers. In this problem when K approaches N the problem apparently becomes the hardest, which may be considered as a special type of phase transition (our unreported experiment shows a sharp increase in complexity as K gets closer to N, followed by a sharp decrease as K>N moves away from N, with the expected symmetry on both the phases). We also intend to study these phenomena closely in the future. Another future direction is to apply different search algorithms replacing MC and FC. We want to study how hybridization improves efficiency with the algorithms faster than MC and FC. In order to find the optimal K we need more experimentation with some real data sets (e.g., from the DIMAC center's dataset). We also intend to increase the size of the studied problems.

Acknowledgment: NSF has partly supported this work.

References

1. Anderson, D.R., Sweendy, D.J. (eds): Statistics. West Publishing Company, (1986).
2. Bayardo, Ur.R., Schrag, R.: Using look-back techniques to solve exceptionally hard SAT instances, in Proc. CP-96 (1996) 46-60.
3. Brassard, G. and Harvey, W.: Algorithmics – Theory and Practice, Englewood Cliffs, NJ, Prentice Hall (1988).
4. Dechter, R.: Enhancement Schemes for Constraint Processing: Backjumping, Learning, and Cutset Decomposition, Artificial Intelligence 41 Vol 3 (1990) 273-312.
5. Glover, F. and Laguna, M.: Tabu Search, Kluwer Academic Publishers, 1997.
6. Haralick, R.M. and Elliott, G.L.: Increasing Tree Search Efficiency for Constraint Satisfaction Problems. Artificial Intelligence 14 Vol. 3 (1980) 263-313.
7. Hogg, T. and Williams, C.: Solving the Really Hard Problems with Cooperative Search, in Proc. of AAAI93 (1993) 231-236.
8. Jussien, N. and Lhomme O.: Local Search with Constraint Propagation and Conflict-based Heuristics, Artificial Intelligence. 139 (2002) 21-45.
9. Kondrak, G. and Beek, P. van: A Theoretical Evaluation of Selected Backtracking Algorithms. Artificial Intelligence Journal 89 (1997) 365-387.
10. Minton, S., Philips, A., Johnston, M.D., and Laird P.: Minimizing Conflicts: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems, Journal of Artificial Intelligence Research 1 (1993) 1-15.
11. Pargas, R.P. and Ludwick, J. (eds): Hybrid Search Algorithms. Proceedings of the ACM Symposium on Applied Computing, San Jose, California (1997) 269-273.
12. Pesant, G. and Gendreau, M.: A View of Local Search in Constraint Programming. Springer, Berlin (1996) 353-366.
13. Richards, E.T. and Richards, B.: Non-systematic Search and Learning: An Empirical Study. in Proc. Conference on Principles and Practice of Constraint programming, Pisa (1998) 370-384.
14. Schaerf, A.: Combining Local Search and Look-Ahead for Scheduling and Constraint Satisfaction Problems. Proc. IJCAI-97, Nagoya, Japan (1997) 1254-1259.
15. Selman, B., Levesque, H., Mitchell, D.: A New Method for Solving Hard Satisfiability Problems, in Proc. AAAI-92, San Jose, CA (1992) 440-446.
16. Shaw, P.: Using Constraint Programming and Local Search Methods to solve Vehicle Routing Problems, Proc. 4th Conference on Principles and Practice of Constraint Programming. Pisa (1998) 417-431.
17. Sumitaka, A.: Explicit Solutions of the N-Queens Problem, SIGART, 2 Vol. 2 (1991).
18. Tsang, E.: Foundations of constraint Satisfaction, Academic Press Ltd., London, (1993).
19. Williams, C.P. and Hogg T.: Exploiting the Deep Structure of Constraint Problems, Artificial Intelligence 70 (1994) 73-117.
20. Zhang, J.: Automatic Symmetry Breaking Method Combined with SAT. Proceedings of the ACM Symposium on Applied Computing, LasVegas, Nevada (2001) 17-21.