Experiments of Intelligent Algorithms on Ramsey Graphs

Jihad Mohamad AlJaam
Department of Computer Science and Engineering, Qatar University, Qatar

Abstract: *Ramsey numbers are known to be hard combinatorial problems that have many important applications including number theory, algebra, geometry, topology, set theory, logic, ergodic theory, information theory, and theoretical computer science. The evaluation of Ramsey numbers using intelligent algorithms has been extensively studied in the last decades and only few numbers are currently known. Almost all of these methods failed to find the exact value of Ramsey numbers as they are over constraints problem. They have succeeded only to improve some upper and lower bounds of these numbers. In this work, we have tested the following intelligent algorithm: Backtracking, local search, tabu search and simulated annealing on some extremely hard instances of Ramsey numbers namely R (5, 9) - 120 and R (6, 8) - 121. As we failed to solve these hard instances using the previous techniques, we decided to combine them together in a hybrid metaheuristic algorithm and succeeded to generate the expected solutions. This new hybrid algorithm seems efficient and promising. It can be applied also on different combinatorial problems even if deep mathematical properties of the problems' domain are not on hand.*

Keywords: *Metaheuristic*, *hybrid algorithms, optimization*, *combinatorics*, *Ramsey numbers*.

Received November 19, 2005; accepted April 5, 2006

1. Introduction

Ramsey-type theorems have roots in different branches of mathematics and computer science and the theory developed from them influenced such diverse areas as number theory, set theory, geometry, ergodic theory, complexity, algorithms, parallelism, logic, and networking. Ramsey F. P. (1930) stated his fundamental theorem in a general setting and applied it to formal logic. Vera Rosta, published in [27] an excellent paper about the applications of Ramsey numbers. The finite version of this theorem is as follows. For all t, n, k in N, there exists R in N so that, for $m \geq R$, if the k-tuples of a set M of cardinality m are t-colored, then there exists M' included in M of cardinality n with all the k-tuples of M' having the same color. In other simplified terms, Ramsey-type theorems are showing that if a large enough system is partitioned arbitrarily into finitely many subsystems, at least one subsystem has a particular property. Also, every irregular structure, if it is large enough, contains a regular substructure of some given size and thus total disorder is impossible. Erdös and Szekeres defined the Ramsey numbers in an elegant way using graph terms as follows. For the graphs G_1, G_2, …, G_t, the graph Ramsey number $R(G_1, G_2, \ldots, G_t)$ is the smallest integer R with the property that any complete graph of at least R vertices (i. e., graph of order R) whose edges are partitioned into t color classes contains a monochromatic sub-graph isomorphic to G_i in the i-th color for some i, $1 \leq i \leq t$. We talk about classical Ramsey numbers or simply Ramsey numbers when all G_i graphs are complete graphs. These numbers have many applications in different fields, for instance in theoretical computer science, to obtain lower bounds for parallel sorting, in information theory, applications of Ramsey theory mostly involve finding maximal independent sets for various graphs, which correspond to information channels meaning that to obtain a lower bound on the capacity of unions of channels, constructing lower bounds of Ramsey numbers were used, while density results in number theory are essential in harmonic analysis applications.

Evaluating Ramsey numbers has been extensively studied since 1955, and only few numbers are currently known as their evaluation problems are effectively over constrained problems and their space searches are extremely large. Many heuristic algorithms have been proposed to solve combinatorial problems in general and to determine the values of Ramsey numbers in particular [1, 13, 14, 17, 18, 19, 28, 31], and almost all of them failed to find the exact values of these numbers. They have succeeded only to associate with particular Ramsey numbers, some upper and lower bounds. The following table shows the known values till now.

Table 1. Known non-trivial values of binary Ramsey numbers since 1955 till 2005.

R (3, 3)	R (3, 4)	R (3, 5)	R (3, 6)	R (3, 7)
6	9	14	18	23
R (3, 8)	R (3, 9)	R (4, 4)	R (4, 5)	R (3, 3, 3)
28	36	18	25	17

In this work, we have tested some well-known heuristic search techniques on some extremely hard instances of Ramsey numbers namely R (5, 9) - 120

and R (6, 8) - 121. These methods are, local search, simulated annealing, tabu search and backtracking. As we couldn't solve the previous hard instances with the mentioned techniques, we combined them together in a meta-heuristic algorithm and succeeded to generate the expected solutions. This hybrid algorithm seems efficient and promising. It can be adapted easily to solve different combinatorial problems even if deep mathematical properties of the problems' domain are not on hand.

The paper is organized as follows. Section 2 presents the intelligent algorithms that we have tested on hard instances of Ramsey numbers. Section 3 discusses our hybrid metaheuristic algorithm, while section 4 discusses our experiments. Finally, section 5 concludes the paper.

2. Intelligent Algorithms

In this section, we discuss and present the following intelligent algorithms: Greedy algorithm, local search, simulated annealing, tabu search and backtracking.

2.1. Optimization Problem and Search Space

An optimization problem is generally formulated as follows:

minimize or *maximize f (x)*
subject to x in *D*

We call *f* the *objective function*, *D* the *feasible region* that satisfies all the given constraints, and a solution *x* in *D* a *feasible solution*. If *D* has combinatorial features, then the problem is called a *combinatorial optimization problem*. The set of solutions of an optimization problem, which may be potentially visited in a local search algorithm, is called the *search space*. If generating feasible solutions is relatively easy (like the traveling salesman problem [9]), then we may define the feasible region *D* as the whole search space.

On the other hand, if generating feasible solutions is not easy (i. e., searching solutions in *D* is difficult, like the graph coloring problem [20]), we may take the problem structures into account and define an appropriate search space *D**. If we adopt a search space different from *D*, we need also to modify the objective function *f* to *f**, so that we can evaluate the amount of unfeasibility of given solutions as well. Consequently, the appropriateness of *D** and *f** depends on the structure of the given problem.

2.2. Greedy Algorithm

The greedy method is a one-path algorithm that constructs a feasible solution or a pseudo-solution step by step, on the basis of the local effectiveness. The algorithm should have a list of possible candidates, a predicate *solution* to test whether a given set of candidates give a solution (not necessarily optimal), a predicate *feasible* to test if a set of candidates can be extended to a solution (not necessarily optimal), a selection function, *select*, which chooses a candidate which has not yet been used. The algorithm is presented in Figure 1.

```
Greedy (C: Set of Candidates)
Begin
  S = {}
  While (Not Solution (S)) and (C ≠ {}) do
   Begin
     x = SelectCandidat (C)
     C = C \ {x}
     If FeasibleSolution (S U {x})
       Then S = S Union {x}
   End
   Return S
End
```

Figure 1. Greedy algorithm.

2.3. Local Search

The general idea of local search approach consists of performing a search by iteratively changing a complete assignment of variables. For each iteration, a neighborhood of potential successor states is considered. A state's quality can be computed by a cost function called objective function. The basis for the selection of a successor state is mostly the expected improvement with respect to the current state's costs. So, the central issue in local search is the transition from one state to the successor state. As it is quite uncertain what kind of change improves a current state, a whole bunch of neighbor states may usually analyzed to find the next successor state. However, There are no general rules to follow, the kind of neighborhood and successor selection being a heuristic matter. The term heuristic implies the existence of some domain knowledge for guidance. Many successful applications of local search gain their power from sophisticated updates rather than from re-computations. For this purpose, additional structures may be maintained. Figure 2 shows the process of the local search algorithm.

To select which variable to change its value in each step, the effect of changing a variable's value is assessed. Note that, changing the value of a variable, may make some unsatisfied constraints satisfied, and some satisfied constraints unsatisfied. The numbers of constraints that will be made unsatisfied by changing a variable's value is called the break-count of the variable at the current assignment. Local search algorithms attempt to change a variable's value with zero break-count, trying to make the next assignment no worse than the current one. To find a variable with zero break-count, a local search algorithm first selects an unsatisfied constraint C, uniformly randomly, from all unsatisfied constraints. This is called constraint pick. If there is a variable of zero break-count, local search then picks such a variable, uniformly randomly,

from the ones that qualify (called flat pick). If no zero break-count variable exists in C, local search then makes a random choice. With probability *p* it chooses, uniformly randomly, a variable from all the variables involved in C (called noise pick); or with probability *1 - p* it selects a variable with the least break-count, breaking a tie arbitrarily if multiple choices exist (called greedy pick). The algorithm takes three parameters to run: Number of tries, maximal number of flips in each try, and a probability for noise pick, which is commonly referred to as the noise ratio of the algorithm.

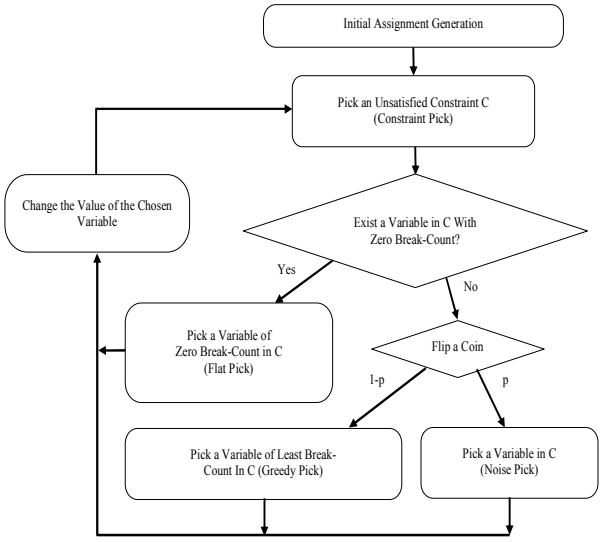

Figure 2. The Process of local search algorithm.

The simple local search algorithm starts from an initial solution or configuration *x*, generated randomly or by a greedy algorithm, and repeat replacing it with a better solution *x'* (i. e., $f(x') < f(x)$) in its neighborhood $N(x)$ until no better solution is found in $N(x)$, where $N(x)$ is a set of solutions obtainable from *x* by slight perturbations. The algorithm is presented in Figure 3.

```
LocalSearch()
Begin
   x = Initial Solution
   Repeat
      x' = SelectSolution(N (x))
      If (f (x') < f (x)) then x = x'
   Until Stopping Criterion is met
   Return x
End
```

Figure 3. Simple local search algorithm.

The main problem of heuristics methods is *local optima*. In fact, when searching for a solution of a given optimization problem, a local search algorithm may often get into a region of the search space containing poor solutions that cannot be improved whatever the number of iterations carried out. This region is called *local optima zone*. To overcome this problem and escape from poor locally solutions, many metaheuristic algorithms have been proposed. Those metaheuristics include random multi-start local search [24], genetic algorithm [7, 23], simulated annealing [2], tabu search [13], and so on. Among variants of the previous methods, are genetic local search [9], greedy randomized adaptive search procedure [8], guided local search [29, 32, 33], ant systems [5] and so on. These metaheuristic algorithms are based on the iteration of the following two steps:

1. Search new solutions on the basis of the previous history.
2. Evaluate the solutions generated in Step 1, and extract necessary information for the future search.

Therefore, metaheuristics can be considered as the collection of ideas of how to use the search history to generate new solutions and how to extract the necessary information from the generated solutions.

From a theoretical point of view, the use of the previously mentioned heuristics methods has not yet been justified. For example, a few convergence theorems for simulated annealing and tabu search exist [6] but they are useless in practice. These theorems simply state that the search has a very high probability of ending with an optimal solution if a disproportionate computing time is allowed (larger, in fact, than the time needed for a complete enumeration of the solution space). Practically, these heuristics methods are very competitive. In this race for competitiveness, the most efficient methods hybridize two or more heuristic algorithms as each one may help the others in the resolution process as we will show in this work.

2.4. Simulated Annealing

The simulated annealing method was proposed by Kirkpatrick *et al*. in 1983 to solve the traveling salesman problem [21]. The method is a variant of the local search method, in which test solutions are randomly chosen from the neighborhood $N(x)$, and accepted with probability that is 1 if the test solution is better than the current one *x*, and positive probability even if the test solution is worse than *x* (only in the beginning of the search). By assigning a positive probability to a move to a worse solution, the search is normally able to avoid getting into poor locally solutions' zone. The probability is controlled by a parameter *t* called *temperature*, whose idea stems from the physical annealing process. Temperature *t* is set to an empirical large value in the beginning of the search to allow the acceptance of worse neighborhood solutions with a high probability. The temperature is gradually decreased as the search proceeds to reduce the probability of acceptance of worse neighborhood solutions. When $t = t_0$, only better neighborhood solutions are accepted and the behavior of the algorithm becomes the same as that of simple local search algorithm. The algorithm is shown in Figure 4.

```
SimulatedAnnealing()
Begin
  x = Initial Solution
  t = Initial Value
  Repeat
    x' = SelectSolution (N (x)) randomly
    Δ  = f (x') – f (x)
    If (Δ ≤ 0) then x = x'
    Else
     accept x' with a certain
          probability e^(-Δ/t)
     Decrease t
  Until Stopping Criterion is met
  Return x
End
```

Figure 4. Simulated annealing algorithm.

2.5. Tabu Search

The tabu search is a metaheuristic method proposed by Glover [13] in 1986. The overall approach is to avoid getting into cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the search space previously visited (hence tabu). So, the main idea of the method is to locally and repeatedly modify a solution while memorizing these modifications to avoid visiting the same solutions a second time or in a cyclic manner. Consequently, modifications are stored in a short (or eventually long, depending of the treated problem) list T, called *tabu list*, which forbids their use for a certain number of iterations. Basically, at each iteration the best solution in the neighborhood $N(x) \setminus (\{x\} \cup T)$ is chosen as the next possible candidate solution. At initialization, the goal is to make a coarse examination of the solution space, known as *diversification*, but as candidate locations are identified the search is more focused to produce local optimal solutions in a process of *intensification*. A basic tabu search algorithm is shown in Figure 5.

```
Generate an initial solution x, and initialize the list T
  While a Stopping Criterion is not met Do
    Begin
      x' = Select Best Solution
      N (x) \ ({x} Union T)
      If (f (x') – f (x) ≤ 0) then x = x'
      Update T
    End
```

Figure 5. Simple tabu search algorithm.

2.6. Backtracking Algorithm

Backtracking technique has long been used as a strategy for solving combinatorial hard problems and has been extensively studied [4, 11, 12, 22, 26, 30]. The method is a refinement of the brute force approach[1], which systematically searches for a solution to a problem among all available options. It does so by assuming that the solutions are represented by vectors $(v_1, ..., v_m)$ of values and by traversing the domains of the vectors, in a depth first manner, until the solutions are found. When invoked, the algorithm starts with an empty vector. At each stage, it extends the partial vector with a new value. Upon reaching a partial vector $(v_1,..., v_i)$ which cannot represent a partial solution, the algorithm backtracks by removing the trailing value from the vector, and then proceeds by trying to extend the vector with alternative values. The algorithm is shown in Figure 6.

```
TrySolve (v1,...,vi)
Begin
  If (v1,...,vi) is a solution
    Then return (v1,...,vi)
  For each v do
    If (v1, ..., vi, v) is acceptable vector then
    Begin
      solution = TrySolve (v1, ..., vi, v)
      If solution ≠ {}
        Then return solution
    End
  End
End
```

Figure 6. Backtracking algorithm.

If S_i is the domain of v_i, then S_1 x … x S_m is the solution space of the problem. The validity criteria used in checking for acceptable vectors determines what portion of that space needs to be searched, and so it also determines the resources required by the algorithm. The traversal of the solution space can be represented by a depth-first traversal of a tree. The tree itself is rarely entirely stored by the algorithm in discourse; instead a path toward a root is stored, to enable the backtracking. The backtracking algorithm may be improved by some filtering techniques, which aim at pruning the search space in order to decrease the overall duration of the search. Many new backtracking algorithms have been proposed to solve successfully some combinatorial problems [22, 25]. However, when the size of the problem instance is relatively large, backtracking methods become totally inefficient due to the exponential nature of the algorithms.

3. Hybrid Metaheuristic Algorithm

In this section, we present and discuss the hybrid metaheuristic optimization algorithm that combines tabu search, simulated annealing, and backtracking algorithms to solve combinatorial hard problems in general and Ramsey numbers instances in particular.

The algorithm starts by generating an initial solution x of the treated problem using a greedy procedure. The solution quality is then improved repeatedly by a local search procedure, in which, we use a FIFO list T of short length (i. e., a tabu list of n locations, with $5 \leq n \leq 8$) to forbid visiting the same region of the search space in a cyclic manner. At each iteration, either the best solution in the neighborhood $N(x) \setminus (\{x\} \cup T)$

[1]The *brute force* approach for any combinatorial problem is to enumerate all possible solutions, testing each in turn and rejecting those, which fail to meet the required conditions.

or a random solution (not in *T*) is chosen as the next possible solution according to the value of the choice variable which, in turn, selected randomly. To avoid getting rapidly into a poor locally solutions' zone, we accept moves to worse solutions, only in the beginning of the search. For this purpose, we use two parameters *t* and threshold that should be carefully tuned to get good performance. The value of *t* is gradually reduced as the search proceeds to decrease the probability of acceptance of worse solutions. The stopping criterion is met either whenever $f(x') \geq f(x)$ for all x' in $N(x)$, or after a certain number of iterations without improving the quality of the current solution. At this stage, we use a systematic search algorithm to try to improve the quality of the current solution. The algorithm is shown in Figure 7.

```
Generate an initial solution x using a greedy algorithm
Initialize the tabu list T, and the parameters, t and the threshold

Repeat
   choice = random(1)
   If (choice == 0) then
    x' = SelectSolution (N (x))
       (randomly with x' not in T)
   Else
      x' = BestSolution (N (x)) (for all
      x' in N (x) and x' not in T)
   If (f (x') ≤ f (x)) then x = x'
   Else accept x' if t > threshold
     Update the FIFO tabu list T
     Decrease t.
Until Stopping Criterion is met
If Solution is not yet found and
  (Stopping criterion ≠ expected
  solution) then
     Begin
        Construct a partial solution xp by removing from x all
        variables in conflict v1, v2, ..., vk
        Sort v1, v2, ..., vk according to their degree of conflicts
        Use a backtracking algorithm to place correctly v1, v2,
        ..., vk
     End
```

Figure 7. Hybrid metaheuristic algorithm.

4. Experiments

We have tested all the previous algorithms on some extremely hard edge-coloring Ramsey graphs problems namely R (5, 9) - 120 and R (6, 8) - 121 using the heuristics given in our research paper [17]. We succeeded only to solve these instances using the hybrid meta-heuristic algorithm, and failed to get a solution using a backtracking, a local search, a tabu search, and a simulating annealing alone.

Note that, $R(k_1, k_2) - n$, denotes a complete graph of order *n*, for which it exists a bi-coloring of its edges (let's say red and blue) that doesn't contain neither a monochromatic red clique of order k_1 nor a monochromatic blue clique of order k_2.

Our experimental shows that local minima from local search techniques (simulated annealing and tabu search) reside close to one another, forming clusters in configuration landscapes. Thus, a backtracking technique at this stage may play an important role to reach the expected solution. In fact, the local minima reached by the algorithm must share many parts of the solution structures with the optimal solutions. If we extract some of the structure information from the local minima, we can then use it to adjust the local search in such a way that it attempts to fix the parts of the current state which are not compatible with optimal solutions, so as to guide the search toward the regions of the search space containing high quality solutions.

When implementing the hybrid algorithm, one must determine, the strategy of generating the initial solution *x*, the objective function *f*, the neighborhood solutions $N(x)$, the search space, the stop criterion, the length of the tabu list *T*, and the values of the parameters *t* and threshold. In our case, for constructing simple Ramsey graphs of the following instances R (5, 9) - 120 and R (6, 8) - 121 we have used the following:

- *Initial Solution*: The initial solution consists of coloring the edges of the simple complete graph K_n using a greedy algorithm that minimizes the number of monochromatic blue cliques of order k_1, and the number of monochromatic red cliques of order k_2.
- *Objective Function*: The objective function *f* is the sum of blue monochromatic cliques of order k_1, and the number of red monochromatic cliques of order k_2.
- *Neighborhood*: Two simple Ramsey graphs are neighbors Iff we can obtain one of them from another by changing only the color of one edge.
- *Stopping Criterion and Tabu List*: The stopping criterion is met whenever $f(G) = 0$. The tabu list *T* of short length keeps the indices of recently modified edges.
- *Space Search*: The space search is the set of all colorable Ramsey graphs.
- *Parameters t, Threshold*: The values of these two parameters are adjusted experimentally with each iteration.

4.1. Heuristics

A problem of fundamental interest and practical importance is how to utilize problem structural information, in a search algorithm to cope with the high computational cost of difficult problems, as well as to improve the performance of the algorithm. Thus much research is needed to exploit problem structural information in order to demonstrate the viability of incorporating such information in search algorithms. One of the challenges in utilizing structural information of a problem, in a search algorithm, is to make the algorithm not only work on random problem instances, but also perform well on individual problem instances, especially those from real-world applications.

Let G be a simple Ramsey graph of order n, then it exists a bi-coloring (red, blue) of the edges of G, in which, all edges of the same length have the same color, such coloring is called *cyclic coloring* or *symmetric coloring*. The length of the edge $\{v_i, v_j\}$ is equal to:

$$|i - j|,$$
$$If\ |i - j| < [n / 2]$$
$$or\ to$$
$$|i - j + n|,\ otherwise.$$

So, the lengths of the edges of G vary from 1 to [n / 2].

Table 2 shows the solution of the Ramsey numbers instances R (5, 9) - 120 and R (6, 8) - 121.

Table 2. Solutions for the Ramsey numbers instances R (5, 9) - 120 and R (6, 8) - 121.

R (5, 9) - 120	Blue Edges	2, 3, 6, 7, 13, 15, 17, 18, 19, 20, 22, 23, 28, 29, 31, 33, 41, 42, 43, 45, 48, 52, 53, 54, 60
	Red Edges	1, 4, 5, 8, 9, 10, 11, 12, 14, 16, 21, 24, 25, 26, 27, 30, 32, 34, 35, 36, 37, 38, 39, 40, 44, 46, 47, 49, 50, 51, 55, 56, 57, 58, 59
R (6, 8) - 121	Blue Edges	2, 4, 5, 7, 11, 12, 13, 14, 15, 18, 20, 28, 30, 31, 34, 35, 40, 41, 44, 46, 47, 48, 49, 53, 56, 60
	Red Edges	1, 3, 6, 8, 9, 10, 16, 17, 19, 21, 22, 23, 24, 25, 26, 27, 29, 32, 33, 36, 37, 38, 39, 42, 43, 45, 50, 51, 52, 54, 55, 57, 58, 59

5. Conclusion

We have tested some well known intelligent and efficient algorithms on extremely hard instances of Ramsey numbers, namely R (5, 9) - 120 and R (6, 8) - 121 without succeeding to find the solutions due to the extreme combinatorial search spaces of these instances where the numbers of attempts is between 2^{5886} and 2^{7381}. We have then proposed and implemented a hybrid metaheursitic algorithm that uses a tabu search and simulated annealing in their simplest way along with a backtracking technique and we generated successfully the expected solutions of these hard instances. The algorithm is flexible and can be adapted to solve efficiently other combinatorial hard problems like the satisifiability problem, the multiple sequence alignment, the traveling salesman problem, etc.

References

[1] Aarts E. H. L. and Lenstra J. K., *Local Search in Combinatorial Optimization*, Wiley, 1997.

[2] Aarts E. H. L., Korst J. H. M., and Van Laarhoven P. J. M., "Simulated Annealing," in Aarts E. H. L., and Lenstra J. K. (Eds), *Local Search in Combinatorial Optimization*, Wiley, pp. 91-120, 1997.

[3] Bollobás B., "Extremal Graph Theory," in Graham R. L., Grotshel M., and Lovasz L. (Eds), *Handbook of Combinatorics*, Volume II, MIT Press, Cambridge, Mass., 1995.

[4] Butler G. and Lam C. W. H., "A General Backtrack Algorithm for the Isomorphism Problem of Combinatorial Objects," *Journal of Symbolic Computation*, vol. 1, no. 4, pp. 363-381, 1985.

[5] Colorni A., Dorigo M., and Maniezzo V., "Distributed Optimization by Ant Colonies," *in Proceedings of the First European Conference on Artificial Life (ECAL-91)*, pp. 134-142, 1991.

[6] Faigle U. and Kern W., "Some Convergence Results for Probabilistic Tabu Search," *ORSA Journal on Computing*, vol. 4, pp. 32-37, 1992.

[7] Falkenauer E., *Genetic Algorithms and Grouping Problems*, Wiley, 1999.

[8] Feo T. A. and Resende M. G. C., "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, vol. 6, pp. 109-133, 1995.

[9] Freisleben B. and Merz P., "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems," *in Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 616-612, 1996.

[10] Garey M. R. and Johnson D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[11] Gerhart S. L. and Yelowitz L., "Control Structure Abstraction of the Backtracking Programming Technique," *IEEE Transaction on Software Engneering*, SE 2, pp. 285-292, 1976.

[12] Ginsberg M. L., "Dynamic Backtracking," *Journal of Artificial Intelligence Research*, vol. 1, pp. 25-46, 1993.

[13] Glover F. and Laguna M., *Tabu Search*, Kluwer Academic Publishers, 1997.

[14] Gu J., "Efficient Local Search for Very Large-Scale Satisfiability Problems," *SIGART Bulletin*, vol. 3, no. 1, pp. 8-12, 1992.

[15] Graham R., Rothschild B. L., and Spencer J. H., *Ramsey Theory*, Wiley, 1990.

[16] Hattingh J. H. and Henning M. A., "Bipartite Ramsey Theory," *Utilitas Math.*, vol. 53, pp. 217-230, 1998.

[17] Jaam J. M., "Ramsey Numbers by Stochastic Algorithms with New Heuristics," in Deza, Euler R., and Manoussakis I. (Eds), *Combinatorics and Computer Science*, Selected Paper, LNCS 1120, pp. 161-181, 1995.

[18] Jaam J. M., Fliti T., and Hussain D., "New Bounds of Ramsey Numbers Via a Top-Down Algorithm," *in Proceedings the International Workshop on Studying and Solving Really Hard Problems*, pp. 110-118, 1995.

[19] Jaam J. M., "Coloriage Cyclique pour les Hypergraphes Complets Associes Aux Nombres de Ramsey Classiques Ternaires," *Bulletin of Symbolic Logic*, vol. 1, no. 2, pp. 241-242, 1995.

[20] Jensen T. R. and Toft B., *Graph Coloring Problems*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, 1995.

[21] Kirkpatrick S., Gelatt C. D., and Vecchi M. P., "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[22] Kondrak G. and Van Beek P., "A Theoretical Evaluation of Selected Backtracking Algorithms," *Artificial Intelligence*, vol. 89, pp. 365-387, 1997.

[23] Muhlenbein H., "Genetic Algorithms, Local Search in Combinatorial Optimization," in Aarts E. H. L. and Lenstra J. K. (Eds), Wiley, pp. 137-171, 1997.

[24] Papadimitriou C. H. and Steiglitz K., *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, 1998.

[25] Prosser P., "Hybrid Algorithms for the Constraint Satisfaction Problem," *Computational Intelligence*, vol. 9, no. 3, pp. 268-299, 1993.

[26] Roever W. P., "On Backtracking and Greatest Fixpoints, in Formal Description of Programming Constructs," in Neuhold E. J. (Ed), North-Holland, pp. 621-636, 1978.

[27] Rosta V., "Ramsey Theory Applications, *The Electronic Journal of Combinatorics*, DS13, pp. 1-40, January 2005.

[28] Selman B., Kautz H., and Cohen B., "Local Search Strategies for Satisfiability Testing," in Johnson D. S. and Trick M. A. (Eds), *in Proceedings of the 2nd DIMACS Challange on Cliques, Coloring, and Satisfiability*, vol. 26, pp. 521-532, 1996.

[29] Voudouris C. and Tsang E., "Guided Local Search and its Application to the Traveling Salesman Problem," *European Journal of Operational Research*, vol. 133, pp. 469-499, 1999.

[30] Walker R. J., "An Enumerative Technique for a Class of Combinatorial Problems, in *Proceedings of Symposia in Applied Mathematics 10*, 1960.

[31] Walser J. P., "Solving Linear Pseudo-Boolean Constraint Problems with Local Search," *in Proceedings of the Fourteenth National Conference on Artificial Intelligence* (AAAI-97), pp. 269-274, 1997.

[32] Zhang W., Rangan A., and Looks M., "Backbone Guided Local Search for Maximum Satisfiability," *in Proceedings of IJCAI-03*, Acapulco, Mexico, pp. 1179-1184, 2003.

[33] Zhang W., "Configuration Landscape Analysis and Backbone Guided Local Search, Part I: Satisfiability and Maximum Satisfiability," *Artificial Intelligence*, vol. 158, pp. 1-26, 2004.

Jihad Mohamad Al Jaam obtained his Bachelor degree in 1998, a Master degree in 1990, and a PhD degree in 1994 in computer science and mathematics of computing from France South Universities and the National Council of Scientific Research (CNRS), France. Currently, he is an associate professor of computer science and engineering (and candidate to a full professorship rank) at the College of Engineering, Department of Computer Science and Engineering, Qatar University. He is the author and co-author of 12 national textbooks in information technology and around 70 research papers published in different international scientific journals and conferences proceedings. His research interests include stochastic algorithms, artificial intelligence, logic and mathematics of computing, satisfiability, problem solving, graph theory, combinatorics, and information retrieval.