Meta-heuristics for Robust Graph Coloring Problem

Andrew Lim and Fan Wang *
Dept. of Industrial Engineering and Engineering Management,
Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong
*Corresponding author, email: fanwang@ust.hk

Abstract

In this paper, the Robust Graph Coloring Problem (RGCP), an extension of the classical graph coloring, is solved by various meta-heuristics. After discussing the search space encoding and neighborhood structure, several meta-heuristics including genetic algorithm, simulated annealing and tabu search are developed to solve RGCP. The experimental results on various sizes of input graph provide the performance of these meta-heuristics in terms of accuracy and run time.

1. Introduction

The graph coloring problem is a well-known NP-hard problem, which has numerous applications in the real engineering and business world [7]. The goal of the graph coloring problem is to use the minimal number of colors to color the vertices of a given graph, with the constraint that a pair of adjacent vertices must receive different colors. Since it has been proved that the graph coloring problem is an NP-hard problem [4], a lot of heuristic algorithms have been proposed, such as tabu search based algorithm [2], simulated annealing based algorithm [6] and evolutionary based algorithm [1].

The Robust Graph Coloring Problem (RGCP), is a widely used extension in uncertainty management from the classical graph coloring problem [9]. RGCP focuses on building robust coloring for a given graph by a fixed number of colors, taking into consideration the possibility of penalizing those coloring where both vertices of an missing edge having the same color. The penalty function depends on the application domain.

Although RGCP is an extension of classical graph coloring, RGCP has its own specifications in many characteristics. We have presented a genetic algorithm based on the partition-based neighborhood for solving the RGCP [8]. In this paper, we study some other meta-heuristics such as simulated annealing and tabu search. All the meta-heuristics including genetic algorithm are compared in terms of accuracy and running time, for various sizes of graph.

2. Problem Statement

The RGCP can be defined as follows: Given the graph $G = (V, E)$ with $|V| = n$, a positive integer c and a penalty set $P = \{p_{ij}, (i,j) \in \overline{E}\}$, the objective function of RGCP is to find

$$\min R(G) \equiv \sum_{(i,j)\in\overline{E}, C(i)=C(j)} p_{ij} \qquad (1)$$

where C is a coloring mapping, i.e., $C : V \rightarrow 1, 2, \cdots, c$ satisfying $C(i) \neq C(j), \forall (i,j) \in E$. Any RGCP instance is characterized by (G, c, P). Depending on various application domains, the penalty set may have different definitions.

Since the NP-hardness of RGCP has been proved in [9], the above binary programming method can only solve very small instances optimally in acceptable computing time.

3. A Case Study - Crew Assignment Problem

In this section, we will introduce a case study of RGCP application - robust crew assignment problem for a regional airline. *Hong Kong Dragon Airlines* is a reginal airline that serves over 30 passenger destinations across Asia. We consider the instance of four crew teams for five round-trip domestic flight routes from Hong Kong to other Asian cites, illustrated in Table 1.

In Table 1, columns 1 to 4 are the identity number (ID), the flight route, the departure time from the home base airport HKG (Hong Kong International Airport) and the return time to HKG in the same day, respectively.

The constraints for crew assignment problem are (1) each flight route should be operated by one and only one crew team; (2) each crew team can operate at most one flight

Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)
1082-3409/04 $20.00 © 2004 **IEEE**

ID	Route	Departure	Return
1	HKG-PEK-HKG	09:00	16:00
2	HKG-BKK-HKG	09:00	14:00
3	HKG-PEK-HKG	15:00	22:00
4	HKG-SIG-HKG	16:00	23:00
5	HKG-SIG-HKG	08:00	15:00

Table 1. An Instance of One-day Timetable of Flight Route

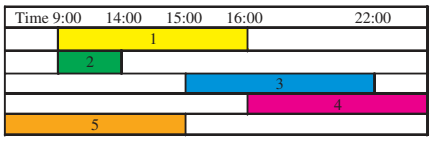

Figure 1. Overlap Relationship Based on the Timetable of Table 1

route at any time; (3) each crew team should have at least 60 minutes or more in HKG to switch from one flight route to another; (4) a crew team may be asked to take one more flight routes due to insufficient personnel. It is clear that the most important relationship is the overlap of flying duration. The overlap relationship based on Table 1 is shown in Figure 1.

We can use a graph to model the overlap relationships where each vertex represents one flight route. An edge exists between two vertices if a crew member cannot work on the two flight routes represented by the two vertices. This means that the end time of the earlier route must be earlier than the start time of the later route by at least 60 minutes. For instance, based on the overlap relationship in Figure 1, the corresponding graph model is showed as Figure 2.

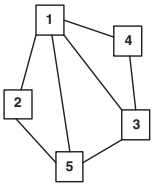

Figure 2. Graph Model Corresponding the Timetable in Table 1

The crew assignment problem can be modelled as a graph coloring problem where each vertex represents a flight route and each color represents one crew team. For the graph in Figure 2, it is easy to see that the minimum number of colors needed is 3. This means that only 3 crew teams can completely handle all 5 flight routes without violating any constraint. One possible optimal coloring for the graph in Figure 2 is: $[C(1) = 1, C(2) = C(3) = 2, C(4) = C(5) = 3]$(denoted as "Solution 1") where $C(i) = j$ means arranging crew team j to flight route i.

Next, let us consider flight delays which occur very often. For example, due to the bad weather in Bangkok (BKK), flight route No.2 has been delayed and returns to HKG at 16:00 instead of the original schedule at 14:00. This change results in the change of the overlap relationship between flight routes No.2, No.3, and No.4. The modified graph model is shown in Figure 3. In this case, three colors are not enough. However, if the previous coloring solution for the graph of Figure 2 is: $[C(1) = 1, C(2) = 2, C(3) = 3, C(4) = C(5) = 4]$(denoted as "Solution 2"), it obviously handles this uncertain flight delay without any crew assignment rescheduling.

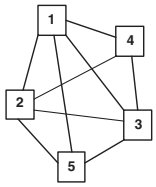

Figure 3. Graph Model If Flight Route No.2 Delays to Return at 16:00

In many applications, we prefer to achieve a robust coloring solution rather than the one with the minimum number of colors. In the robust crew assignment problem, the definition of 'Robustness' takes into account the uncertainty of the flight delays which is the norm in day-to-day flight operations.

One simple way to model the uncertainty is to estimate the probabilities of the presence of missing edges. We use the following equation which considers the overlap rate between two overlapped flight routes No.i and No.j,

$$h_{ij} \equiv \frac{T_{tr}}{\max\{T_i^{dep}, T_j^{dep}\} - \min\{T_i^{ret}, T_j^{ret}\} + \alpha}$$

where T_i^{dep} and T_i^{ret} represent the departure and the return time for flight route No.i, T_{tr} means the minimal ground transfer time. The unit of these three parameters is in minute. α is a constant. From the above equation, we obtain the following uncertainty matrix H which presents all uncertain changes for the fixed timetable of Table 1 (when

α is set to 10)

$$H = \begin{bmatrix} * & - & - & - & - \\ & * & 0.86 & 0.46 & - \\ & & * & - & - \\ & & & * & 0.86 \\ & & & & * \end{bmatrix} \quad (2)$$

In addition, we formalize the objective function for a given graph topology $G = (V, E)$ with given uncertain information matrix H,

$$\min R(G) \equiv \sum_{(i,j)\in\overline{E}, C(i)=C(j)} h_{ij}. \quad (3)$$

For Solution 1, the value of robustness is $R = p(2,3) + p(4,5) = 0.86 + 0.86 = 1.72$. On the other hand, for Solution 2, $R = p(4,5) = 0.86$. From the above calculations, we say that Solution 2 is more robust than Solution 1.

4. Search Space

We use partition-based encoding for search space encoding, where a set of vertices belonging to a class will be assigned the same color. In other words, a solution can be present as $\{V_1, V_2, \cdots V_c\}$, where $V_i = \{j|C(j) = i, 1 \leq j \leq n\}, 1 \leq i \leq c$. For instance, we can represent the Solution 1 for the graph in Figure 2 as $\{\{1\}, \{2,3\}, \{4,5\}, \Phi\}$ when $c = 4$. For Solution 2, its partition-based encoding is $\{\{1\}, \{2\}, \{3\}, \{4,5\}\}$. It is definite that partition-based encoding can represent any coloring solutions, feasible or unfeasible.

For neighborhood construction, we have the operator namely Single Vertex Color Modification. The operator first randomly selects one vertex $v_i (1 \leq i \leq n)$ among all n vertices in the graph. Then, the new color of v_i is assigned a fixed or random color, e.g. $C_{new}(v_i) = c', c' \neq C(v_i)$ where c' is given by randomization or determination.

5. Meta-heuristics

5.1. Simulated Annealing

Our simulated annealing algorithm for solving RGCP is proposed as Algorithm 1. After the initial solution generation, the iteration is repeated until the stop criterion is satisfied that the current temperature is smaller than the fixing minimum temperature T_{min}. In each iteration, a new solution $coloring'$ from the neighborhood of the current solution $coloring$ is selected. If the new solution is better than the current best solution in terms of the objective function value, the current best solution is updated. Otherwise, different from the local search, the new solution $color'$ is accepted by comparing the value of $exp((R(coloring) - R(coloring'))/T)$ with a random number generated from a uniform distribution on the interval [0,1). Furthermore, the algorithm goes into the cooling schedule that decreases the temperature from T to $T * ControlFactor$ where the parameter $ControlFactor$ ($0 < ControlFactor < 1$) controls the speed of convergence of the algorithm.

Algorithm 1 Simulated annealing for solving RGCP

```
coloring ← Call Initial Solution Generation;
T ← T_max;
L ← max{n/MinIteration, MinIteration};
repeat
  for i = 1 to L do
    coloring' ← N(coloring);
    if R(coloring') < R(coloring) then
      coloring ← coloring';
    else
      if exp((R(coloring) − R(coloring'))/T) > random[0, 1) then
        coloring ← coloring';
      end if
    end if
  end for
  T = T * ControlFactor;
until T ≤ T_min
return color
```

5.2. Tabu Search

In our tabu search algorithm for solving RGCP, we define two tabu lists: $TabuVec$ and $TabuList$. $TabuVec$ is a one dimension list which defines the tabu memory of the coloring action of each vertex. If the current iteration index t is smaller than the value of $TabuVec[v_i]$, the vertex v_i is tabued to change color in this iteration. $TabuList$ is a two dimension list which defines the tabu memory of the coloring action of each color assigned to each vertex. In other words, if the current iteration index t is smaller than the value of $TabuList[v_i, c_j]$, the vertex v_i is tabued to be assigned by color c_j in this iteration. The tenure of tabu list $TabuVec$ is a random number generated from a uniform distribution on the interval [1,$MaxTenure$), while the tenure of $TabuList$ is from a uniform distribution on [3, $MaxTenure$). Here, we set the longer tenure for $TabuList$ rather than $TabuVec$, because keeping the color assignment for a vertex has more impact for the diversity of solutions than just tabuing a fixed color to that vertex.

Tabu search begins from an initial solution, proceeding iteratively from one solution to another under neighborhood construction until the iteration time has exceeded the fixed the maximum iteration time $tMax$. During each iteration, if a better solution is found, the current best solution will be updated and corresponding coloring action will be tabued. We propose two tabu search algorithms for solving RGCP. One is based on the greedy neighborhood construction, namely "greedy tabu search"; the other is based on the complete neighborhood construction namely "complete tabu search". In both of them, the neighborhood of single vertex color modification is applied.

The difference between the greedy tabu search and the complete tabu search exists in finding the maximum reduction of $R(G)$ for the single vertex color modification (denoted as $\triangle R(v', c')$ when the untabued vertex v' is assigned as a new color c'). In the greedy tabu search(illustrated as Algorithm 2), we split the calculation into two parts, e.g. $\triangle R(v', c') = decR(v') + incR(v', c')$. Here, $decR(v')$ is the reduction of $R(G)$ if the current color $color[v']$ of v' is removed. And $incR(v', c')$ is the reduction of $R(G)$ if c' is assigned to the uncolored vertex v'. So, in the greedy tabu search, seeking the maximum $\triangle R(v', c')$ is implemented by two steps: first we find the vertex v' which has the maximum $decR(v')$. Then the optimal color c' is obtained for vertex v' to have the maximum $incR(v', c')$. In contrast, the complete tabu search (illustrated as Algorithm 3) constructs the whole neighborhood of the single vertex color modification in each iteration. It obtains the maximum $\triangle R(v', c')$ directly by enumerating all combinations of vertex and colors.

Algorithm 2 Greedy tabu search for solving robust coloring

$TabuVec[v_i] \leftarrow 0, \forall v_i \in V$;
$TabuList[v_i, c_i] \leftarrow 0, \forall v_i \in V, c_i \in C$;
$TabuTenureVec = random(1, MaxTenure), TabuTenureList = random(3, MaxTenure)$;
$coloring \leftarrow$ Call Initial Solution Generation;
$t, t' \leftarrow 0$;
repeat
  $t' \leftarrow t' + 1$;
  $coloring' \leftarrow coloring$;
  $v' \leftarrow max^{-1}_{v_i \in V, t' > TabuVec[v_i]} decR(v_i)$;
  $c' \leftarrow max^{-1}_{c_i \in C, t' > TabuList[v', c_i], color'[v_j] \neq c_i \forall (v', v_j) \in E} incR(v', c_i)$;
  $coloring'[v'] = c'$;
  if $R(coloring') < R(coloring)$ **then**
    $coloring \leftarrow coloring'$;
    $t \leftarrow 0$;
    $TabuVec[v'] = t' + TabuTenureVec$;
    $TabuList[v'c'] = t' + TabuTenureList$;
  else
    $t \leftarrow t + 1$;
  end if
until $t > tMax$
return $coloring$;

Algorithm 3 Complete tabu search for solving robust coloring

$TabuVec[v_i] \leftarrow 0, \forall v_i \in V$;
$TabuList[v_i, c_i] \leftarrow 0, \forall v_i \in V, c_i \in C$;
$TabuTenureVec = random(1, MaxTenure), TabuTenureList = random(3, MaxTenure)$;
$coloring \leftarrow$ Call Initial Solution Generation;
$t, t' \leftarrow 0$;
repeat
  $t' \leftarrow t' + 1$;
  $coloring' \leftarrow coloring$;
  $\triangle R' \leftarrow 0$
  for all v_i such that $v_i \in V$ and $t' > TabuVec[v_i]$ **do**
    $\triangle R^* \leftarrow max_{c_l \in C, t' > TabuList[v_i, c_l], color'[v_j] \neq c_l \forall (v_i, v_j) \in E} \triangle R(v_i, c_l)$, where c^* is the color corresponding to $\triangle R^*$;
    if $R^* > R'$ **then**
      $v' \leftarrow v_i, c' \leftarrow c^*$;
      $\triangle R' \leftarrow \triangle R^*$;
    end if
  end for
  $coloring'[v'] = c'$;
  if $R(coloring') < R(coloring)$ **then**
    $coloring \leftarrow coloring'$;
    $t \leftarrow 0$;
    $TabuVec[v'] = t' + TabuTenureVec$;
    $TabuList[v', c'] = t' + TabuTenureList$;
  else
    $t \leftarrow t + 1$;
  end if
until $t > tMax$
return $coloring$;

Method	Meta-Heuristics
HGA-E	Hybrid GA(Enumerative Search)
HGA-R	Hybrid GA(Random Search)
SA	Simulated Annealing
TS-G	Tabu Search(Greedy)
TS-C	Tabu Search(Complete)

Table 2. Summary of Methods for Solving RGCP

6. Experimental Results

Four sizes of test data are designed to evaluate the performance of various meta-heuristics in different sizes of graph: Small Size ($n = 10, 15, 20$), Middle Size ($n = 50, 100$), Large Size ($n = 250, 500$) and Huge Size ($n = 1000$). There are 15 test sets in total, 7 sets for Small Size, 3 sets for Middle Size, 4 sets for Large Size and 1 set for Huge Size. For each test set, we have randomly generated 50 instances where the missing edge penalties are generated with the uniform distribution in the interval [0,1]. The graph density is fixed to be 0.5. For our experiments, we use a Pentium 4 personal computer with a 1GHz CPU and 256MB RAM.

We tested five heuristic search methods for solving the RGCP under the above test sets. They are illustrated in Table 2. The first two genetic algorithms could be found in our previous work [8].

First, in Table 3, the overall results are shown for Small, Middle, Large and Huge Size. In the mean while, the corresponding running time is shown in Table 4. These computational results are obtained when we give enough time for running the different methods, where the stopping criteria is that the search is finished when there is no improvement on $R(G)$ for five continual iterations.

Comparing the heuristics (HGA-E, HGA-R, SA, TS-G and TS-C), we can see that for Small Size instances, HGA-R outperforms others. The second best method in accuracy for Small Size is HGA-E. However, their running time is much higher than TS-G and TS-C. From Middle Size, TS-C becomes the best solution with fast running time. It is also the best method for both Large Size and Huge

n	c	HGA-E	HGA-R	SA	TS-G	TS-C
10	4	3.03	2.96	3.23	3.82	3.12
10	5	1.51	1.48	1.69	2.59	1.72
15	5	7.02	6.33	6.84	7.89	7.07
15	6	4.10	3.73	4.30	6.00	4.38
20	5	14.92	13.12	14.72	11.99	15.12
20	8	4.06	3.78	4.64	7.32	4.84
20	10	1.71	1.48	2.03	3.91	1.94
50	18	10.90	8.38	10.57	20.79	10.08
100	35	23.34	15.45	19.65	41.97	16.34
100	50	2.78	1.56	5.30	8.20	2.34
250	70	86.90	48.80	48.90	127.60	54.53
250	90	30.07	17.63	21.5	62.00	20.17
500	150	88.71	62.09	57.00	138.00	64.00
500	250	7.93	21.83	6.90	14.65	2.35
1000	400	43.11	51.07	23.12	62.15	22.86

Table 3. Overall Computational Results ($R(G)$)

n	c	HGA-E	HGA-R	SA	TS-G	TS-C
10	4	0.17	0.31	0.27	0.01	0.03
10	5	0.2	0.27	0.26	0.02	0.00
15	5	0.26	0.3	0.11	0.00	0.00
15	6	0.22	0.28	0.19	0.02	0.02
20	5	0.16	0.18	0.11	0.00	0.03
20	8	0.26	0.38	0.21	0.02	0.01
20	10	0.26	0.36	0.34	0.01	0.02
50	18	0.58	0.6	0.34	0.08	0.13
100	35	0.77	0.82	0.54	0.13	0.94
100	50	9.73	10.68	7.24	9.10	1.37
250	70	22.74	23.52	28.01	16.81	4.05
250	90	31.10	27.65	32.16	30.21	3.74
500	150	108.16	79.06	95.65	64.20	72.72
500	250	150.13	196.03	196.02	156.78	166.45
1000	400	710.02	889.81	800.90	952.24	1178.34

Table 4. Overall Average Computational Time Per Instance (in second)

Size. However, it is time consuming when the size of graph increases. For example, it costs 1178.34 seconds for the set of Huge size when $n = 1000$. SA and HGA-E have good performance for Large Size and Huge Size in accuracy. For instance, for the set of Huge Size, HGA-E obtains $R(G) = 43.11$ by running 710.02 seconds and SA obtains $R(G) = 23.12$ by running 800.90 seconds.

Moreover, to have sufficient comparison on the performance of various heuristics in the same encoding and neighborhood (HGA-E, HGA-R, SA, TS-G and TS-C), we tested the $R(G)$ v.s. the average running time. For Middle Size, HGA-R and HGA-E have the best performance. TS-C obtains the minimum $R(G)$ when the running time is small. For Large Size, TS-C gets accurate $R(G)$ with short running time. However, when n is increased to 500, the performance of SA becomes the best. SA outperforms all other meta-heuristics, where it achieves accurate $R(G)$ by short running time. For instance, for the set $(n, c) = (500, 150)$, TS-C spends 72.72 seconds to get $R(G) = 64$. In contrast, SA uses only 35.60 seconds to get $R(G) = 57.98$. For the Huge Size, TS-C obtains the best $R(G) = 22.86$. However, TS-C is time consuming to obtain this result. On the other hand, SA obtains $R(G) = 26.49$ by only 35 seconds, which saves over 97% running time.

7. Conclusions

In this paper, we have proposed several meta-heuristics for solving Robust Graph Coloring Problem, an interesting extension of classical graph coloring. Local search, simulated annealing and two tabu search methods have been presented and tested by various sizes of data in terms of the accuracy and the running time. The experimental results that we could select right meta-heuristic when we solve the RGCP problem on different sizes of input.

Acknowledgement

We are grateful to Ying Kong for his hard work on coding and experiments.

References

[1] Costa, D., Hertz, A., Dubuis, O. 1995. Embedding a sequential procedure within an evolutionary algorithm for coloring problems. *Journal of Heuristics* 1 105-128.

[2] Dorne, R., Hao, J.K. 1998. Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization, *Chapter 6: Tabu search for graph coloring T-colorings and set T-colorings*. Kluwer Academic, 77-92.

[3] Galinier, P., Hao, J.K. 1999. Hybrid evolutionary algorithm for graph coloring. *Journal of Combinatorial Optimization*. 3 379-397.

[4] Garey M.R., Johnson, D.S. 1979. *Computer and Intractability*. Freeman, San Francisco.

[5] Halldorsson, M.M. 1990. A still better performance guarantee for approximate graph coloring. Technical Report 91-35, DIMACS, New Brunswick, NJ.

[6] Johnson, D.S., Aragon C.R., McGeoch, L.A., Schevon C. 1991. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3) 378-406.

[7] Pardalos, P.M., Mavridou, T., Xue, J. 1998. The Graph Coloring Problems: A Bibliographic Survey. *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers. 2 331-395.

[8] Wang, F., Kong, Y., Lim, A., Guo, S.S. Robust graph coloring problem: A hybrid genetic algorithm. *submitted to Discrete Applied Mathematics*. Sept. 2003.

[9] Yanez, J. Ramirez, J. 2003. The robust coloring problem. *European Journal of Operational Research*. 148(3) 546-558.