A New Neighborhood Based on Improvement Graph for Robust Graph Coloring Problem

Songshan Guo^{+}, Ying Kong^{+}, Andrew Lim*, and Fan Wang$^{*,@}$

$^{+}$Dept. of Computer Science, Zhongshan (Sun Yat-sen) University, Guangzhou, China
*Dept. of Industrial Engineering and Engineering Management, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
fanwang@ust.hk

Abstract. In this paper, we propose a new neighborhood structure based on the improvement graph for solving the Robust Graph Coloring Problem, an interesting extension of classical graph coloring. Different from the traditional neighborhood where the color of only one vertex is modified, the new neighborhood involves several vertices. In addition, the questions of how to select the modified vertices and how to modify them are modelled by an improvement graph and solved by a Dynamic Programming method. The experimental results clearly show that our new improvement graph based k-exchange cycle neighborhood improves the accuracy significantly, especially for large scale heuristic search.

1 Introduction

The graph coloring problem is a well-known NP-hard problem, which has numerous applications in the real engineering and business world [18]. 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 [9], a lot of heuristic algorithms have been proposed, such as the greedy coloring algorithm [15], successive augmentation [2], tabu search based algorithm [6], simulated annealing based algorithm [11] [4] and evolutionary based algorithm [5]. Furthermore, the well-known second DIMACS challenge benchmarks have also been set up to compare different problem solving methods [12].

The Robust Graph Coloring Problem (RGCP), is a widely used extension in uncertainty management from the classical graph coloring problem, which was first introduced in [19]. 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

[@] Corresponding Author

© Springer-Verlag Berlin Heidelberg 2004

color. The penalty function depends on the application domain. One case study of RGCP application, namely "airline crew robust assignment" motivated from an airline, is presented in [14].

We have presented a genetic algorithm to solve the RGCP [14]. In that paper, the major contribution is the new effective partition based encoding method and the genetic algorithm. Different with that paper, in this paper, we focus on creating effective neighborhood structures. A new improvement graph based neighborhood structure is presented, comparing with the traditional operator where we just modify the color of a single vertex every time. A local search algorithm is then developed to compare the performances of such two neighborhoods. From the experimental results for various sizes of graph, the new improved graph based neighborhood obtains better accuracy.

This paper is organized as follows: in Section 2, the RGCP is stated formally. In Section 3, the encoding method of search space is discussed. The two neighborhood structures are then presented in Section 4. We develop a local search algorithm in Section 5 and provide the experimental results in Section 6. Finally, Section 8 presents the conclusions.

2 Problem Statement

The RGCP can be defined formally 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

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

where C is a coloring mapping, i.e., $C : V \to 1, 2, \cdots, c$ 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 (Yanez 2003), the above binary programming method can only solve very small instances optimally in acceptable computing time.

3 Search Algorithm

3.1 Encoding

A partition approach is applied to the search space encoding, where a set of vertices belonging to a class will be assigned the same color [14]. In other words, a solution can be present as $\{V_1, V_2, \cdots V_c\}$, where $V_i = \{j|C(j) = i, 1 \leq j \leq n\}, 1 \leq i \leq c$. It is definite that partition based encoding can represent any coloring solutions, feasible or unfeasible.

3.2 Neighborhood 1: Single Vertex Coloring Modification

The first method for neighborhood construction is 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.

3.3 Neighborhood 2: Improvement Graph

We first define the improvement graph transformed from a given graph with a fixed coloring mapping. For a given graph $G = (V, E)$ and a coloring mapping $C, C : V \to 1, 2, ..., c$, we define the improvement graph $G' = (V', E')$ as follows:

1. $V' = V$;
2. $(i, j) \in E'$, iff $C(i) \neq C(j)$;
3. The weight of the directed edge (i, j), $w(i, j)$, is defined as the reduction (positive or negative) of $R(G)$ when we empty the color of vertex i and change the color of vertex j from $C(j)$ to $C(i)$.

Then, we define the "k-exchange cycle" in an improvement graph G': a k-exchange cycle is a simple cycle in G', which consists of exact k successive edges, e.g. $\{(s_i, e_i)|(s_i, e_i) \in E', \forall 1 \leq i \leq k\}$ satisfying that (1) $(s_i, e_i) \neq (s_j, e_j), \forall 1 \leq i, j \leq k, i \neq j$; and (2) $s_1 = e_k, s_i = e_{i-1} \forall 1 < i \leq k$. We call the sum of the weights of all k edges along the cycle the weight of the k-cycle, e.g. $\sum_{\forall (s_i,e_i) \in EC} w(s_i, e_i)$. By a k-exchange cycle in an improvement graph, we have a new neighborhood operator for the coloring mapping of the corresponding graph, where we $C(e_i) = C(s_i)$ for all i from 1 to k. In the mean while, the reduction of $R(G)$ is equal to the weight of the exchange cycle. It is easy to know that the advantage of this new neighborhood is that the color distribution K is kept unchanged. On the other hand, compare with the previous "single vertex coloring modification", this new operator can affect more vertices in the same operation.

Since there are a lot of k-exchange cycles in an improvement graph, we need to find the optimal one with the maximum reduction of $R(G)$. A Dynamic Programming (DP) method is developed to find the optimal k-exchange cycle.

A few denotations are first defined:

p_k: a path, e.g. the consequence of k vertices $v_{i1}, v_{i2}, ..., v_{ik}$ covers the edges (v_{i1}, v_{i2}), $(v_{i2}, v_{i3}) \cdots$ and $(v_{i(l-1)}, v_{ik})$.

$Length(p_k)$: the length of the path p_k, e.g. k

$cycle(p_k)$: the cycle corresponding to the path p_k, e.g. the cycle along v_{i1}, $v_{i2}, ..., v_{ik}, v_{i1}$

$w(cycle(p_k))$: the total reduction of $R(G)$ along the cycle p_k. In other words, it is the sum of the weights of all edges belonging to $cycle(p_k)$, e.g. $w(cycle(p_k)) = \sum_{(v_i,v_j) \in cycle(p_k)} w(v_i, v_j)$

$s(p_k)$: the first vertex of the path p_k, e.g. v_{i1}

$e(p_k)$: the last vertex of the path p_k, e.g. v_{ik}

$p_k + v$: the new path with an added vertex v at the end of the original path p_k.

Based on the definition of the k-exchange cycle, the following DP formula is applied to obtain the best k-exchange cycle, where the p_k represents the path of the best k-exchange cycle, $cycle(p_k)$.

$$p_{k+1} = max^{-1}_{p_k+v}\{w\left(cycle(p_k + v)\right)|\forall v \in \Omega\} \text{ for all } p_k, k \geq 2 \qquad (2)$$

where $\Omega = \{v|v \in V(G') \text{ and } (e(p_k), v) \in E(G') \text{ and } C(v_i) \neq C(v_j)\, \forall v_j \in p_k\}$

We illustrate the DP algorithm for seeking the best k^*-exchange cycle as Algorithm 1. Here, k^*-exchange cycle is marked as the best cycle among all h-exchange cycles where h is from 1 to k. The DP constructs the best exchange cycle in the order of length k. First, all single edges with negative weight are added into the search candidate list - *List*. They are considered as all possible best paths with length $k = 1$. Then, the best paths with length from 2 to k are obtained by iteration based on the DP equation(2). During the DP iteration, the best path with the maximum reduction of $R(G)$ is remarked. Finally, the best k^*-exchange cycle is determined.

Since the "longest path problem" is NP-complete [9], the computational complexity should be exponential if the above DP search covers the whole search space. To balance the solution accuracy with the running time, we create a candidate list management scheme (Algorithm 2) similar to the Beam Search. We set *List* to be a sorted doubly linked list where the elements represent the candidate paths in decreasing order of w. In each time a new candidate path is found, it will be inserted into *List* by sort. There is an importation parameter to control the maximum size of the *List*, $MaxListLength$. Once the length of the *List* exceeds the fixed maximum length, the last element of the *List* will be removed to keep the length. If the $MaxListLength$ is big enough, the DP can guarantee to produce the optimal solution for the k^*-exchange cycle. On the other hand, a small $MaxListLength$ may lose the optimal solution. However, it can reduce the search space efficiently.

3.4 Local Search

We illustrate the local search algorithm for solving the RGCP in Algorithm 3. The basic idea of the local search is that it starts from an initial solution and repeatedly replaces it with a better solution in its neighborhood until a better solution could not be found in the neighborhood structure.

Algorithm 3 Local search for solving RGCP

```
1: $coloring \leftarrow$ Call Initial Solution Generation;
2: $Improvement \leftarrow true$; remarks if there is a new better solution found
3: repeat
4:    construct the neighborhood structure $N(coloring)$ of $coloring$;
5:    if find any solution $coloring'$ from $N(coloring)$ so that $R'(coloring) < R(coloring)$ then
6:       $coloring \leftarrow coloring'$;
7:    else
8:       $Improvement \leftarrow false$;
9:    end if
10: until Not $Improvement$
11: return $coloring, R$;
```

Algorithm 1 DP Algorithm for find best k-exchange cycle

```
1: $List \leftarrow \phi$;
2: $w_{best} \leftarrow 0$;
3: for all $(v_i, v_j)$ such that $(v_i, v_j) \in E(G'), w(v_i, v_j) > 0$ do
4:     $Add(List, (v_i, v_j))$;
5:     if $w(cycle(v_i, v_j)) > w_{best}$ then
6:         $w_{best} = w(cycle(v_i, v_j))$
7:     end if
8: end for
9: for $l = 1$ to $k - 1$ do
10:    $p \leftarrow Pop(List)$;
11:    for all $v$ such that $(e(p), v) \in E(G')$ and $w(p + v) > 0$ and $color[v_j] \neq$
       $color[v] \forall v_j \in p$ do
12:        $p' \leftarrow p + v$;
13:        if $(v, s(p)) \in E(G')$ and $w(cycle(p')) > w_{best}$ then
14:            $w_{best} \leftarrow w(cycle(p'))$;
15:            $p_{best} \leftarrow p'$;
16:        end if
17:        if $Length(p) \leq k - 1$ then
18:            $Add(List, p')$;
19:        end if
20:    end for
21: end for
22: return $cycle(p_{best}), w_{best}$;
```

Algorithm 2 Algorithm of candidate list management - $Add(List, p)$

```
1: Insert $p$ into $List$ in the decreasing order of $w(cycle(p))$;
2: if $Length(List) > MaxListLength$ then
3:     delete $List[Length]$;
4: end if
```

4 Experimental Results

4.1 Test Data And Experimental Environment

We have designed four sizes of test data 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.

4.2 Comparison of Neighborhood Structures

In Table 1, the performance comparison between single vertex color modification and improvement graph based k-exchange cycle are provided for Small, Middle, Large and Huge Size, in terms of accuracy ($R(G)$) and running time. 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.

Table 1. Comparison on neighborhood structure

		single vertex color modification		k-exchange cycle	
n	c	$R(G)$	Runing Time	$R(G)$	Running Time
10	4	3.67	0.00s	2.96	0.00s
10	5	2.36	0.00s	1.54	0.02s
15	5	7.63	0.01s	6.62	0.01s
15	6	5.68	0.01s	3.99	0.01s
20	5	15.92	0.00s	14.22	0.01s
20	8	7.15	0.02s	3.81	0.02s
20	10	3.86	0.01s	1.46	0.06s
50	18	18.62	0.02s	7.28	0.37s
100	35	32.79	0.14s	10.87	1.04s
100	50	8.40	3.00s	1.55	0.45s
250	70	93.98	5.23s	44.27	9.57s
250	90	51.18	6.58s	15.39	7.59s
500	150	113.50	68.02s	52.37	88.37s
500	250	23.67	69.65s	1.27	78.79s
1000	400	61.33	552.34s	18.40	687.19s

For the above results, it is clear that the new improvement graph based neighborhood outperforms the single vertex color modification. Especially for Large Size and Huge Size, the new neighborhood obtains much better accuracy. For instance, for the case $(n, c) = (1000, 400)$, the new neighborhood achieves $R(G) = 18.40$, improving 70% relatively. In addition, the running time of such two neighborhood are in the same level.

Configuration of k-Exchange Cycle. As presented in Section 4, a DP method is applied to find the best k-exchange cycle under several configurations including k setting and the management scheme of the candidate list. In Table 2, the performance of the local search with k-exchange cycle are illustrated where k is set from 3 to 7 with the sort candidate list management (the maximum length is 10). The corresponding running time is shown in Table 3. The note $k - sort/unsort - MaxListLength$ remarks one configuration.

From the results of performance *vs.* k, running time increases with the increase of k slowly. The performance in accuracy for different k is not diverse. Hence, $k = 4$ with the shortest running time is the best choice.

Table 2. Performance of k-exchange cycle with different k on $R(G)$

n	c	3-sort-100	4-sort-100	5-sort-100	6-sort-100	7-sort-100
10	4	2.96	2.96	2.96	2.96	2.96
10	5	1.55	1.54	1.54	1.54	1.54
15	5	6.62	6.62	6.62	6.62	6.62
15	6	4.01	3.99	3.93	3.96	3.96
20	5	14.22	14.22	14.22	14.22	14.22
20	8	4.04	3.81	3.85	3.85	3.83
20	10	1.56	1.46	1.46	1.46	1.46
50	18	7.85	7.28	7.35	7.14	7.13
100	35	11.99	10.87	10.73	10.56	10.56
100	50	1.68	1.55	1.56	1.53	1.52
250	70	46.97	44.27	43.24	42.12	42.35
250	90	16.17	15.39	14.48	14.29	13.76
500	150	53.86	52.37	49.96	48.30	48.53
500	250	1.11	1.27	1.13	1.01	1.04
1000	400	16.16	18.40	17.44	16.53	15.89

Table 3. Average Running Time Per Instance of k-exchange Cycle with Different k's (in second)

n	c	3-sort-100	4-sort-100	5-sort-100	6-sort-100	7-sort-100
10	4	0.00	0.00	0.00	0.00	0.01
10	5	0.02	0.02	0.02	0.02	0.01
15	5	0.01	0.00	0.00	0.00	0.02
15	6	0.02	0.00	0.00	0.02	0.00
20	5	0.00	0.00	0.00	0.00	0.00
20	8	0.04	0.02	0.02	0.02	0.02
20	10	0.02	0.06	0.05	0.06	0.06
50	18	0.50	0.37	0.53	0.34	0.41
100	35	0.76	1.04	0.90	0.70	1.06
100	50	0.76	0.45	0.51	0.84	0.90
250	70	9.48	9.57	8.07	7.81	7.33
250	90	11.45	7.59	8.47	8.27	7.58
500	150	114.18	88.37	96.11	94.04	78.17
500	250	134.62	78.79	77.01	71.77	61.58
1000	400	2261.62	687.19	725.49	738.55	733.65

To asses the efficiency of the management of candidate list to balance accuracy and running time, in Table 4 and Table 5, accuracy and running time comparison among three management schemes are presented for $k = 4$, including unlimited candidate list (4-unsort), sort candidate list with maximum size of 100 (4-sort-100) and sort candidate list with maximum size of 200 (4-sort-200).

Table 4. Performance of different candidate list management on $R(G)$

n	c	4-unsort	4-sort-100	4-sort-200
10	4	2.96	2.96	2.96
10	5	1.54	1.54	1.69
15	5	6.62	6.62	7.23
15	6	3.99	3.99	4.19
20	5	14.22	14.22	14.22
20	8	3.81	3.81	3.82
20	10	1.46	1.46	1.60
50	18	7.24	7.28	7.32
100	35	10.79	10.87	10.56
100	50	1.51	1.55	1.52
250	70	42.16	44.27	44.33
250	90	14.08	15.39	14.82
500	150	47.71	52.37	49.26
500	250	0.83	1.27	1.08
1000	400	13.44	18.40	15.58

Table 5. Average running time per instance of k-exchange cycle with different candidate list management (in second)

n	c	4-unsort	4-sort-100	4-sort-200
10	4	0.00	0.00	0.03
10	5	0.01	0.02	0.00
15	5	0.00	0.00	0.00
15	6	0.02	0.00	0.02
20	5	0.00	0.00	0.00
20	8	0.02	0.02	0.06
20	10	0.05	0.06	0.26
50	18	0.65	0.37	0.62
100	35	1.36	1.04	1.80
100	50	1.32	0.45	0.33
250	70	13.01	9.57	9.00
250	90	12.51	7.59	11.48
500	150	159.68	88.37	108.61
500	250	148.38	78.79	111.80
1000	400	3010.93	687.19	1452.32

It is clear that the order of running time from the slowest to the fastest is 4-unsort, 4-sort-200 and 4-sort-100. However, the order of accuracy from the best to the worst is also 4-unsort, 4-sort-200 and 4-sort-100. In other words, the larger size of the candidate list (the more running time), the higher accuracy in terms of management scheme of k-exchange cycle.

5 Conclusions

In this paper, we have proposed a new neighborhood structure based on the improvement graph for solving the Robust Graph Coloring Problem, an interesting extension of classical graph coloring. Different from the traditional neighborhood where the color of only one vertex is modified, the new neighborhood involves several vertices. In addition, the questions of how to select the modified vertices and how to modify them are modelled by an improvement graph and solved by a Dynamic Programming method. A local search algorithm has been developed to provide the computational results of performance comparison on various sizes of graph. The experimental results clearly show that our new improvement graph based k-exchange cycle neighborhood obtains much better performance than the traditional neighborhood, especially for large scale heuristic search.

References

1. Barthelemy, J.P. Guenoche, A. 1991. *Trees and Proximity Representations.* John Wiley Sons, New York.
2. Brelaz, D. 1979. New methods to color vertices of a graph. *Communications of ACM* 22 251-256.
3. Chaitin, G.J. 1982. Register allocation and spilling via graph coloring. *SIGPLAN '82 Symposium on Compiler Construction, Boston, Mass. June 1982*, 17 98-105.
4. Chams, M., Hertz, A., Werra, D.de. 1987. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research* 32 260-266.
5. Costa, D., Hertz, A., Dubuis, O. 1995. Embedding a sequential procedure within an evolutionary algorithm for coloring problems. *Journal of Heuristics* 1 105-128.
6. 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.
7. Galinier, P., Hao, J.K. 1999. Hybrid evolutionary algorithm for graph coloring. *Journal of Combinatorial Optimization.* 3 379-397.
8. Gamst, A. 1986. Some lower bounds for a class of frequency assignment problems *IEEE Transactions of Vehicular Technology.* 35(1) 8-14.
9. Garey M.R., Johnson, D.S. 1979. *Computer and Intractability.* Freeman, San Francisco.
10. Halldorsson, M.M. 1990. A still better performance guarantee for approximate graph coloring. Technical Report 91-35, DIMACS, New Brunswick, NJ.
11. 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.
12. Johnson D.S., Trick, M.A. 1996. *Proceedings of the 2nd DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society. 26.
13. Joslin, D.E., Clements, D.P. 1998. Spueaky wheel optimization. *the National Conference on Artificial Intelligence, AAAI-1998*, Edmonton, Alberta, Canada.
14. Kong, Y., Wang, F., Lim, A. Guo, S.S. 2003. A New Hybrid Genetic Algorithm for the Robust Graph Coloring Problem. *Proceeding of Australian Conference on Artificial Intelligence 2003* 125-136.

15. Korte, B., Vygen, J. 2002. *Combinatorial Optimization*. Springer-Verlag, second edition, 2002.
16. Kubale, M., Jackowski, B. 1985. A generalized implicit enumeration algorithm for graph coloring. *Communications of the ACM*. 28 412-418.
17. Opsut, R.J., Roberts, F.S. 1981. On the fleet maintenance, Mobile radio frequency, task assignment and traffic phasing problems. *The Theory and Applications of Graphs*. John Wiley Sons, New York. 479-492.
18. 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.
19. Yanez, J. Ramirez, J. 2003. The robust coloring problem. *European Journal of Operational Research*. 148(3) 546-558.