Journal of the Korean
OR/ MS Society
Vol. 19, No. 3, December 1994

On the k-coloring Problem

TAEHOON PARK and CHAE Y. LEE

Abstract

A fixed k-coloring problem is introduced and dealt with by efficient heuristic algorithms. It is shown that the problem can be transformed into the graph partitioning problem. Initial coloring and improving methods are proposed for problems with and without the size restriction. Algorithms Move, LEE and OEE are developed by modifying the Kernighan-Lin's two way uniform partitioning procedure. The use of global information in the selection of the node and the color set made the proposed algorithms superior to the existing method. The computational result also shows that the superiority does not sacrifice the time demand of the proposed algorithms.

Key words : graph coloring, heuristics, partitioning.

1. INTRODUCTION

The *graph coloring problem* is defined as partitioning the node set of a graph into as few independent sets as possible. It is a well-known combinatorial problem which is shown to be of NP-complete class. A lot of papers [1][3][4][5][7][10] have been devoted to this problem and many applications accomplished with the graph coloring models. As an example, the *timetabling problem* [2][4][5][10] is formulated in terms of graph coloring. In a simple timetabling problem, the objective is to schedule exams in the smallest number of time periods such that no individual is required to participate in two exams simultaneously [10].

In the timetabling problem, even if we can find the smallest number of time periods that satisfy the constraints, there may be cases where such an ideal time table is useless due to the number of available time periods. A natural alternative to this problem is to schedule exams to minimize the number of cases where a student is required to participate in two exams simultaneously.

Another application of graph coloring is the formulation of *FMS(Flexible Manufacturing System*[9]) *design* by Chams al. [1]. In an FMS one produces several types of products which are manufactured on machines each equipped with a number of different tools. The FMS is usually divided into smaller subsystems called *Flexible Manufacturing Cells* or *FMC*. One tries to group the products into FMCs such that the manufacturing of products within each cell requires as many common tools as possible.

In Section 2 a *k-coloring problem* is introduced. Its application to the timetabling and FMS design is decribed. It is shown that the k-coloring problem can be transformed into the graph partitioning problem.

Algorithms to solve the k-coloring problem are developed in Section 3. The *two way uniform partitioning algorithm* by Kernighan and Lin [8] is modified for partitioning nodes into k color sets. Problems with and without the size restriction in each color set are considered. An improving method is also investigated to enhance the performance of the proposed algorithm.

Computational results are discussed in Section 4 by generating random graphs with various densities. Solution qualities and CPU times are compared with other procedures.

2. PROBLEM DEFINITION AND FORMULATION

A k-coloring problem is defined as coloring of all the nodes in a graph with k colors such that the sum of weights on the edges between adjacent nodes with the same color is minimized. Consider the timetabling problem in Section 1. To simplify the problem, suppose there are k time periods(k is much smaller that n) in a week and each course takes only a single time period in the week. Let $col(i)=l$ or $i\in P_l$, if node i is colored with color l. Also let w_{ij} be the weight of edge (i, j). Then the k-coloring problem is formulated as :

$$\text{(P) Minimize } \sum_{col(i)=col(j),\, i<j} w_{ij}$$

$$\text{subject to}$$

$$1\le col(i)\le k \text{ for all } i$$

In timetabling problem $col(i)=l$, if course i is opened at time period l. The edge weight w_{ij} corresponds to the number of students who want to take courses i and j simulataneously. Also in the FMS design the weight w_{ij} is defined as the number of tools required either by product type i or by product type j but not by both. Suppose a fixed number k of FMCs is given. The problem becomes partitioning the product types into k subsets(FMCs) P_1, P_2,…, P_k such

that $f(P_1, P_2, \cdots, P_k)=\sum_{l=1}^{k} \sum_{i,j\in P_l,\, i<j} w_{ij}$ is minimized.

The k-coloring problem is very similar to the graph partitioning problem. The *graph partitioning* is to divide the nodes of a graph into k subsets such that the total cost of the edge cut is minimized. The only differnece between (P) and the graph partitioning problem, is that the objective function of (P) is to minimize the sum of internal weights(each weight on the edge between two nodes in the same cluster) while that of graph partitioning is to minimize the sum of external weights(each weight on the edge between two nodes in different clusters).

More specifically, $P=\{P_l\}_{l=1,2,\cdots}$(each *cluster* P_l is nonempty) is called a *k-way partition of* a graph $G(N, E)$, if $P_1\cup\cdots\cup P_k=N$ and $P_l\cap P_m=\phi$ whenever $l\neq m$. Let $|N|=n$.

Definition 1 A k-way partition is said to be admissible with respect to a given integer p if $|P_l|\leq p$ for each $1\leq l\leq k$.

Definition 2 When n is a multiple of k, i. e. $n=km$ for some integer m, the condition restricting all the clusters of the partition to be of equal size is called the size restriction.

In the partitioning problem, in almost all cses, the *admissibility* is given because of the nonnegativity of the weights. However, in k-coloring problem, the condition is not necessary. The relationship between k-coloring and k-way partition problem is shown in the following proposition.

Proposition 1 The k-coloring problem is equivalent to the following problem:

(P1) Maximize $\sum_{col(i)\neq col(j),\, i<j} w_{ij}$

subject to

$$1\leq col(i)\leq k \qquad \textbf{for all } i$$

proof

The sum of all weights in a graph is $\sum_{i<j} w_{ij}$, which is constant. Since $\sum_{col(i)=col(j),\, i<j} w_{ij}=\sum_{i<j} w_{ij}-\sum_{col(i)\neq col(j),\, i<j} w_{ij}$, minimizing the sum of external weights is equivalent to (P1).□

Further, by changing the signs of all w_{ij}'s, the problem of minimizing the external cost is obtained. Thus if the admissibility is added as constraints, the k-coloring problem is equivalent to the graph partitioning problem.

Now by letting $W(i, l)=\sum_{j\in P_l} w_{ij}$ and $D(i, l)=W(i, l)-W(i, col((i))$, the following propositions hold.

Proposition 2 If node i is moved to P_l, the reduced cost(old cost−new cost) becomes $-D(i, l)$.

proof

Let z be the cost occurred by the weights of all the edges except those between i and all the nodes of $P_{col(a)}$. Then the old cost$=z+W(i, col(i))$ and the new cost$=z+W(i, l)$. Thus, the reduced cost becomes

$W(i, col(i))-W(i, l)=-D(i, l)$. □

Proposition 3 If node i and j of different colors are exchanged, the reduced cost becomes

$g(i, j)=-(D(i, col(j))+D(j, col(i))-2w_{ij})$.

proof

Let z be the cost from the weights of all the arcs except those between all the nodes except those between i and nodes with $col(j)$ and j and nodes with $col(i)$. Then the old cost$=z+W(i, col(i))+W(j, col(j))+w_{ij}$ and the new cost$=z+W(i, col(j))+W(j, col(i))-w_{ij}$. Thus, the reduced cost becomes $-D(i, col(j))-D(j, col(i))+2w_{ij}$. □

3. INITIAL SOLUTION ALGORITHMS AND IMPROVING METHODS

PROBLEMS WITHOUT THE SIZE RESTRICTION

In this section, we introduce an initial coloring algorithm Anne and an improving procedure Move. For problems with the size restriction(equal-sized clusters), three size controlling methods are presented.

SEQUENTIAL COLORING METHOD : ANNE

The main idea of the procedure is to match the most preferable node with the most preferable color. Two values are assigned to each (node, color) pair : *anti* and *pref*. They are defined as follows :

$$anti(i, l)=\sum_{col(j)=l} w_{ij}$$

$$pref(i, l)=\sum_{s\neq l} anti(i, s)-anti(i, l)$$

anti is the penalty paid by node i when colored with color l, while *pref* is node i's relative preference of color l to the other colors. In the algorithm the most preferable (node, color) pair is selected by comparing the *pref* values of all the nodes. That is, node a is colored with l^* which satisfies $pref(a, l^*)=max_{i\in c, 1\leq l\leq k} pref(i, l)$, where C is the set of candidate nodes. After the coloring, *anti* and *pref* values of all the uncolored nodes are updated as follows :

Proposition 4 After node a is colored with l^*, if w_{ia} is nonzero, anti'-and pref'-, the updated anti and pref values, respectively, are as follows:

$anti'(i, l^*)=anti(i, l^*)+w_{ia}$

$pref'(i, l^*)=pref(i, l^*)-w_{ia}$

$anti'(i, l)=anti(i, l)$ *if* $l\neq col(a)$

$pref'(i, l)=pref(i, l)+w_{ia}$ *if* $l\neq col(a)$

proof

Clear from the definition of *anti* and *pref*. □

Algorithm Anne

Step 1 : Let C be the candidate list of uncolored nodes and initially set C=N
Set anti(i, l)=0, pref(i, l)=0 for each i, l ;

Step 2 : Color the node with maximum sum of weights with color l ;
Update pref(i, l), anti(i, l) values for each i in C and each l ;

Step 3 : Find a in C and l* which satisfies
pref(a, l*)=$\max_{(i,l)}$ pref(i, l) ;
Delete a from C ;

Step 4 : Update pref(i, l), anti(i, l) values for
each i in C and each l ;

Step 5 : If C is nonempty
go to Step 3 ;
End If ;

RUNNING TIME ANALYSIS OF ANNE

The time complexity is calculated as follows :

1. Time to update *anti* is the number of the nodes in the candidate list. Hence the total time is $(n-1)+(n-2)+\cdots+1=n(n-1)/2$ and it is $O(n^2)$.

2. Time to update *pref* is k times the number of the nodes in the candidate list, so it is $O(kn^2)$.

3. Time to select the node to color is $kn+k(n-1)+\cdots+k=kn(n-1)/2$, which results in $O(kn^2)$

4. Hence the time complexity is $O(kn^2)$

IMPROVING METHOD : MOVE

After initial coloring of all the nodes with Anne, the partition can be improved by examining the reduced cost of each node. At each iteration the algorithm selects the best (node, color) pair (a, l^*) with the maximal reduced cost and moves the node a to the color set P_{l^*}. Then by Proposition 2, $-D(a, l^*)$ must be maximal, i. e. $D(a, l^*)=\min_{i\in C,\ 1\leq l\leq k} D(i, l)$, where C is the candidate list of nodes. Let the reduced cost from the i-th move be g_i. The problem is then to find m such that $g=g_1+g_2+\cdots+g_m$ is maximized.

Proposition 5 After node a is moved to P_r, D'(i, l), the updated D(i, l) value is updated as follows:

$D'(i, col(a)) = D(i, col(a)) - w_{ia}$ *if* $col(i) \neq col(a)$ **and** $col(i) \neq l^*$ **(5.1)**

$D(i, col(a)) - 2w_{ia}$ *if* $col(i) = l^*$ **(5.2)**

$D'(i, l^*) = D(i, l^*) + w_{ia}$ *if* $col(i) \neq col(a)$ **and** $col(i) \neq l^*$ **(5.3)**

$= D(i, l^*) + 2w_{ia}$ *if* $col(i) = col(a)$ **(5.4)**

$D'(i, l) = D(i, l) + w_{ia}$ *if* $col(i) = col(a)$ **and** $l \neq col(a)$, $l \neq l^*$ **(5.5)**

$= D(i, l) - w_{ia}$ *if* $col(i) = l^*$ **and** $l \neq col(a)$, $l \neq l^*$ **(5.6)**

$= D(i, l)$ ***otherwise*** **(5.7)**

proof

For a node i which is neither with $col(a)$ nor l^*, $D(i, col(a))$ values are decreased by w_{ia} because a is now with l^* and $W(i, col(a))$ value is decreased by the same amount, so(5.1) holds. Since the amount is the same as the increase of $W(i, l^*)$, *(5.3)* also holds.

For a node i with $col(a)$, $W(i, col(a))$ is decreased by w_{ia} and $W(i, l^*)$ is increased by w_{ia}, so (5.4) and (5.5) holds. For a node i with l^*, similar argument shows that (5.2) and (5.6) holds. Otherwise, there is no change in $W(i, l)$ and $W(i, col(i))$, so there is no change of value $D(i, l)$.□

Note in the procedure that a reduced cost g_i may result in positive even if g_{i-1} is negative. Thus, the improving procedure is repeated as far as g is positive by starting from the previous partition. The algorithm stops when $g=0$. The resulting partition is called *the phase 1 optimal solution.*

Algorithm : move

Step 1 : Construct an initial k-way partition P ;

Step 1 : C=N
  Calculate W(i, l), D(i, l) values for each i, l ;
  index=1 ;

Step 2 : Find a in C and l* between 1 to k such that
  D(a, l*)=min D(i, l) ;
  C=C−{a} ;

Step 3 : Update D(i, l) values for each i in C
  and each l ;
  Let $g_{index} = -D(a, l^*)$;

Step 4 : If C is nonempty
  index=index+1 ;
  go to Step 2 ;
  End If ;

Step 5 : Find m such that the total reduced cost is maximum ;

$g_{max}=g_1+g_2+\cdots+g_m$;

Step 7 : If $g_{max} > 0$ then

Move the first m nodes ;

go to Step 1 ;

Else

Exit ;

End If

RUNNING TIME ANALYSIS OF MOVE

Since the algorithm Move repeatedly improve the coloring of nodes, let us define a *pass* to be the operation involved with one cycle of identifying the maximum total reduced cost. Then the computational complexity of a pass is calculated as follows :

1. The initial computation of the D-values is an n^2-procedure because every node pair has to be considered.
2. The time to update D-values is proportional to the number of D-values to be updated. Thus the total updating time in a single pass is calculated as follows : First, each $D(i, l)$ value is changed as in the mutually exclusive cases of (5.1)-(5.6). In(5.1)$-$(5.4) $D(i, l)$ is updated if l is in $P_{col(a)}$ or in P_{l^*}. In (5.5)$-$(5.6) it is updated if l is $col(a)$ or l^*. Because there is no need to update $D(i, col(i))$ values, the number of updated values at the jth iteration is $2(n-j)+(|P^{(j)}_{col(a)}|+|P^{(j)}_{l^*}|)(k-3)$, where $P^{(j)}_{col(a)}$ and $P^{(j)}_{l^*}$ represents $P_{col(a)}$ and P_{l^*} at the jth move, respectively. Since $|P^{(j)}_{col(a)}|+|P^{(j)}_{l^*}|\leq n$, the total updating time for one pass becomes no more than $\sum_{j=1}^{n}(2(n-j)+n(k-3))$, which results in a kn^2-procedure.
3. The number of comparison to determine the first node to move at the first step is kn and the total time becomes $kn+k(n-1)+k(n-2)+\cdots+k$ and which equals to $O(kn^2)$.
4. Hence the total time complexity is $O(kn^2)$ for a pass.

4. PROBLEMS WITH THE SIZE RESTRICTION

SIZE CONTROLLING METHODS : TURN, ORDER, CAPACITY

The algorithms proposed in the previous section are for coloring nodes of a graph with no constraints on the size of each color set. When each color set is restricted to have equal number of nodes it is necessary to modify the proposed algorithms. Three size controlling methods

are proposed to modify the initial coloring algorithm Anne as follows :

TURN

Color 1, 2, 3,···, k, 1, 2,··· in turn. That is, at Step 2 of Anne, fix $l=l^*$ where l^* is increased by 1 at each step(color k is followed by color 1). Every color has the equal chance to be colored with.

ORDER

Define K to be the set of candidate colors. That is, at Step 2 of Anne, consider only the $pref(i, l)$ values such that l is in K. Initially set $K=\{1, 2\cdots, k\}$. After coloring a node with l^*, delete l^* from K. When K is empty, replace it with $K=\{1, 2,\cdots k\}$ again. No two nodes can be colored with the same color in a period of k.

CAPACITY

Define K in the same way as in the method ORDER. Initially set $K=\{1, 2,\cdots, k\}$ and delete l^* from K if the number of nodes colored with l reaches n/k. No more than n/k nodes is colored with the same color.

5. IMPROVING ALGORITHMS ON PROBLEMS WITH THE SIZE RESTRICTION

As in the case of Anne, Move itself cannot be used for problems with the size restriction. Two improving methods Local Extreme Exchange(LEE) and Overall Extreme Exchange(OEE) [11] are developed based on the algorithm Move. At each step of OEE the algorithm simultaneously selects two nodes that maximize the reduced cost when they are exchanged. The algorithm LEE on the other hand selects only one node at a time and chooses the most desirable color to move. Then another node is selected from the set and the two nodes are exchanged. In applying the two methods, slight changes need to be made since the reduced cost is different from that of partitioning. In coloring, smaller D-values are preferred.

6. COMPUTATIONAL RESULTS

Four cases, 4, 5, 8 and 10 coloring problems are tested which are within the range of personal computers. All graphs used in the experiments are randomly generated with various *densities*. Two types of weights are examined : graphs with 0−1 weight(max weight=1) and

those with the uniform integer weight in {1, 5}(max weight=5) for each constructed edge. Practically, given the number n of nodes and the density d, a random real variable x which is uniform on [0, 1] is generated for each pair of nodes i, j. Nodes i and j are joined if and only if $x \leq d$. An integer weight w_{ij} which is uniform on {1, max-weight} is assigned to the edge (i, j).

All the algorithms are run on IBM 80486SX personal computers. They are implemented with C programming language. For problems without the size restriction, algorithms Anne and Move are applied to generate initial and improved solutions respectively. Tables 1-4 show the results of k-coloring without the size restriction. The initial color settings obtained by Anne is compared with those by randomly generated solutions. Obviously, Anne produces far better initial solutions than the random method. However, the effect of initial color setting by Anne is diminished after applying the improving the algorithm Move. As a result, no significant difference is found between the two initial coloring procedures. However, the average CPU seconds by algorithm Anne+Move is less than those by Random+Move. The algorithm Anne thus has the effect of generating good initial coloring and reducing the computation time of the improving method Move.

Table 1. Coloring Without Size Restriction(100 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	Random Initial Cost	Random+Move		Anne		Anne+Move	
					Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	481	4	123	15	0.61	21	0.11	8	0.71
			5	92	0	0.77	4	0.11	2	0.44
			10	45	0	0.49	0	0.22	0	0.28
1	0.5	2469	4	598	400	0.60	412	0.11	393	0.44
			5	460	274	0.82	296	0.16	274	0.82
			10	215	52	1.87	72	0.27	55	1.21
1	0.9	4452	4	1079	949	0.88	967	0.16	945	0.60
			5	849	720	0.99	739	0.22	716	1.21
			10	400	281	1.43	296	0.38	282	1.37
5	0.1	1524	4	377	44	0.44	60	0.06	31	0.72
			5	297	3	1.15	16	0.11	6	0.61
			10	148	0	0.44	0	0.22	0	0.22
5	0.5	7628	4	1770	1110	0.61	1220	0.11	1156	0.60
			5	1363	764	0.82	850	0.17	730	1.38
			10	600	144	1.48	182	0.33	140	1.10
5	0.9	13540	4	3182	2600	0.77	2670	0.17	2601	0.61
			5	2572	1922	0.82	2011	0.22	1913	0.71
			10	1208	663	1.42	731	0.38	658	1.54

Table 2. Coloring Without Size Restriction(200 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	Random Initial Cost	Random+Move		Anne		Anne+Move	
					Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	1968	5	370	71	3.96	115	0.38	69	7.03
			8	203	0	3.41	10	0.61	1	3.96
			10	178	0	2.80	0	0.72	0	0.83
1	0.5	10004	5	1938	1353	4.67	1427	0.61	1346	3.96
			8	1254	636	8.35	717	0.93	660	5.11
			10	958	421	6.54	501	1.15	423	8.62
1	0.9	17385	5	3504	3085	6.04	3134	0.83	3100	2.81
			8	2137	1768	5.00	1827	1.26	1770	5.43
			10	1684	1331	7.52	1380	1.59	1318	8.12
5	0.1	5978	5	1107	160	4.62	257	0.39	173	3.13
			8	662	0	5.58	17	0.55	0	3.84
			10	551	0	2.80	0	0.22	0	0.22
5	0.5	30511	5	5840	3856	2.64	4076	0.60	3793	3.95
			8	3621	1718	5.83	2015	0.93	1732	5.88
			10	2890	1104	8.46	1308	1.15	1144	6.75
5	0.9	53382	5	10450	8604	3.30	8741	0.83	8495	2.81
			8	6476	4666	5.05	4927	1.26	4626	6.31
			10	5056	3371	6.59	3639	1.59	3357	9.06

Table 3. Coloring Without Size Restriction(300 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	Random Initial Cost	Random+Move		Anne		Anne+Move	
					Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	4488	4	1086	492	5.49	555	0.77	465	7.69
			5	861	275	9.01	357	0.88	295	5.38
			10	421	2	10.60	25	1.64	4	10.05
1	0.5	22343	4	5528	4388	12.58	4540	1.16	4373	12.26
			5	4418	3281	15.11	3416	1.43	3315	9.01
			10	2171	1154	25.38	1300	2.53	1162	23.57
1	0.9	40397	4	10003	9262	12.58	9379	1.59	9292	7.14
			5	7969	7259	10.49	7346	1.92	7243	6.42
			10	3939	3238	10.54	3328	3.52	3226	16.15
5	0.1	12925	4	3242	1199	6.98	1439	0.71	1171	6.26
			5	2664	672	9.01	794	0.88	634	12.90
			10	1298	3	10.54	37	1.60	3	7.91
5	0.5	66461	4	16305	12369	16.75	12173	1.21	12344	6.75
			5	12911	9221	9.06	9767	1.37	9133	10.49
			10	6295	3085	21.15	3574	2.52	3056	17.30
5	0.9	121607	4	29719	26579	6.97	26809	1.64	26452	7.19
			5	10450	20457	10.49	20839	1.93	20273	14.01
			10	11601	8600	14.83	9055	3.46	8482	30.87

Table 4. Coloring Without Size Restriction(400 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	Random Initial Cost	Random+Move		Anne		Anne+Move	
					Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	7983	5	1581	612	18.78	736	1.59	626	25.70
			8	978	151	19.94	233	2.26	164	18.96
			10	785	41	26.26	93	2.86	39	21.59
1	0.5	39917	5	7926	6136	18.73	6418	2.47	6201	13.23
			8	8495	3282	26.74	3522	3.74	3246	47.07
			10	3907	2353	22.46	2551	4.51	2379	30.82
1	0.9	71873	5	14221	13142	16.09	13290	3.41	13179	16.81
			8	8822	7763	19.94	7926	5.11	7766	28.40
			10	7009	6017	29.99	6181	6.20	5968	24.93
5	0.1	24205	5	4785	1616	40.26	2016	1.59	1614	17.68
			8	3045	380	29.93	639	2.31	363	32.24
			10	2453	123	22.47	219	2.74	124	17.68
5	0.5	120028	5	23913	17537	26.86	18391	2.48	17575	18.68
			8	14763	9131	23.24	9766	3.73	9054	23.67
			10	11767	6347	52.62	7210	4.55	6418	45.80
5	0.9	215751	5	42794	37055	21.42	38099	3.41	36896	22.19
			8	26557	21217	43.28	21918	5.10	21279	21.80
			10	21279	15931	48.83	18658	6.21	16195	32.41

The k-coloring problem with the size restriction is also examined. The three size controlling methods are first compared as shown in Table 5. The procedure TURN is proved to be the best in the sense of both the solution quality and the time requirement.

Three improving algorithms LEE, OEE and Kernighan-Lin's procedure (KL) are compared for problems with the size restriction. Note in KL that nodes in only two clusters are considered at a time. Algorithm TURN is employed to generate initial coloring in the three procedures. The computatinal results are illustrated in Tables 6-9.

In most cases the algorithm OEE outperforms two other coloring procedures. The relative superiority of the algorithm OEE seems to be independent of the number of nodes, number of colors, and densities of the graph. The CPU time until termination by the OEE are less than those by the KL in most cases. The average solution quality of LEE is very close to that of KL. However the CPU time required in LEE is much less than those in two other methods. The outperformance of the algorithm OEE is considered to be due to the exploitation of overall reduced cost information of nodes in the process of determining the node pair to exchange.

6. CONCLUSIONS

Efficient heuristic algorithms are developed for the k-coloring problems. An initial solution procedure Anne and an improving method Move are developed for problems without the size restriction. Algorithm Move which is modified from the Kernighan-Lin's two way uniform partitioning algorithm controls the k clusters such that the reduced cost is maximized when a selected node is moved into a selected color set. The time complexity of both Anne and Move shows that the computational effort is proportional to the number of colors.

The k-coloring with equal-sized-cluster problem is shown to be equivalent to the graph partitioning problem. To solve the problem with the size restriction three size controlling methods and improving procedures LEE and OEE are applied. The computational result shows the relative superiority of OEE to LEE and KL. From the aspect of computational time demand LEE is proved to be the fastest whose solution quality is very close to that of the algorithm KL.

Table 5. Comparison of Three Size Controlling Methods (400 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	TURN		ORDER		CAPACITY	
				Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	7983	5	776	0.66	1567	1.10	838	1.43
			8	270	0.82	975	1.53	332	2.09
			10	155	0.88	804	1.87	180	2.69
1	0.5	39917	5	6373	1.60	6699	2.03	6509	2.47
			8	3565	2.42	3981	3.07	3539	3.63
			10	2601	2.63	3185	3.62	2750	4.40
1	0.9	71873	5	11316	2.63	13484	3.02	13322	3.35
			8	7917	3.95	8158	4.56	7994	5.05
			10	6173	4.40	6422	5.38	6238	6.16
5	0.1	24205	5	2077	0.66	2517	1.16	2370	1.42
			8	739	0.82	1705	1.59	919	2.20
			10	434	0.88	1740	1.86	563	2.64
5	0.5	120028	5	18584	1.65	18669	2.25	18875	2.37
			8	10056	2.47	10199	3.07	10282	3.57
			10	7359	2.63	7598	3.68	7633	4.40
5	0.9	215751	5	37857	2.59	38126	3.02	38282	3.35
			8	20074	3.96	22246	4.56	22357	5.11
			10	17013	4.39	17179	5.38	17287	6.15

Table 6. Graph Coloring With Size Restriction(100 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	LEE		OEE		KL	
				Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	481	4	15	0.39	12	3.24	16	3.46
			5	6	0.28	1	0.91	3	3.57
			10	0	0.21	0	1.65	0	4.84
1	0.5	2469	4	395	0.55	387	3.85	401	4.18
			5	283	0.39	278	3.68	277	6.75
			10	68	0.71	64	3.85	68	10.44
1	0.9	4452	4	951	0.38	945	4.55	950	3.95
			5	723	0.55	729	3.57	721	5.00
			10	288	0.66	282	5.39	291	10.76
5	0.1	1524	4	58	0.28	41	3.3	38	4.17
			5	12	0.66	9	2.91	12	3.73
			10	0	0.33	0	1.54	0	5.06
5	0.5	7628	4	1128	0.50	1126	3.29	1140	3.79
			5	751	0.44	762	2.86	795	4.94
			10	146	1.09	173	3.02	156	15.11
5	0.9	13540	4	2612	0.50	2579	3.90	2616	4.28
			5	1990	0.72	1896	7.19	1951	6.70
			10	720	0.77	672	5.49	644	20.65

Table 7. Graph Coloring With Size Restriction(200 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	LEE		OEE		KL	
				Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	1968	5	81	2.58	72	27.07	82	27.85
			8	10	2.36	1	31.53	7	38.01
			10	0	2.59	0	17.19	1	35.81
1	0.5	10004	5	1388	1.70	1366	43.28	1375	34.22
			8	673	3.79	664	42.13	678	56.68
			10	435	5.50	431	34.44	461	73.33
1	0.9	17835	5	3111	2.03	3108	27.19	3109	32.51
			8	1773	3.35	1775	36.75	1762	64.54
			10	1356	2.41	1351	28.84	1350	61.03
5	0.1	5978	5	179	3.90	176	43.34	187	41.80
			8	16	2.91	5	37.02	21	39.50
			10	5	1.54	0	23.13	0	37.51
5	0.5	30511	5	3925	2.96	3901	32.46	3924	51.57
			8	1832	4.34	1741	42.13	1870	41.85
			10	1195	5.77	1137	57.78	1173	77.45
5	0.9	53382	5	8555	1.76	8536	27.07	8496	69.38
			8	4710	5.32	4676	63.22	4676	83.00
			10	3526	3.62	3455	57.78	3455	77.66

Table 8. Graph Coloring With Size Restriction(300 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	LEE		OEE		KL	
				Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	4488	4	497	5.33	476	96.62	469	148.69
			5	294	4.78	266	216.13	302	112.82
			10	23	6.70	9	114.91	24	122.92
1	0.5	22343	4	4410	7.25	4395	128.58	4425	90.51
			5	3300	6.92	3320	89.58	3333	115.18
			10	1214	19.45	1176	153.46	1241	269.30
1	0.9	40397	4	9295	5.38	9298	96.39	9294	111.06
			5	7274	5.82	7263	161.59	7257	98.37
			10	3295	7.20	3240	172.86	3260	168.02
5	0.1	12925	4	1215	8.35	1211	80.35	1192	129.95
			5	648	12.64	667	143.91	673	142.87
			10	26	9.40	10	114.90	22	159.88
5	0.5	66461	4	12573	10.82	12501	96.56	12487	177.85
			5	9148	9.89	9122	251.89	9299	122.43
			10	3259	12.25	3180	249.85	3291	246.95
5	0.9	121607	4	26524	6.37	26549	96.18	26445	167.52
			5	20275	13.46	20322	197.57	20380	249.80
			10	8739	14.00	8703	153.68	8765	216.57

Table 9. Graph Coloring With Size Restriction(400 Node Graph)

Max Weight	Density	Weight Sum	Number of Colors	LEE		OEE		KL	
				Cost	CPU Seconds	Cost	CPU Seconds	Cost	CPU Seconds
1	0.1	7983	5	659	8.74	616	340.15	624	348.67
			8	201	17.36	168	392.77	192	361.79
			10	88	13.18	56	317.52	94	318.73
1	0.5	39917	5	6189	15.54	6171	255.07	6206	301.87
			8	3352	16.69	3261	476.61	3336	407.82
			10	2429	17.46	2365	545.36	2437	410.24
1	0.9	71873	5	13150	8.79	13118	511.41	13158	278.36
			8	7796	16.75	7751	305.12	7793	341.91
			10	6011	27.68	6011	317.20	6051	380.41
5	0.1	24205	5	1649	22.57	1622	339.61	1731	390.57
			8	446	17.19	449	261.61	504	359.43
			10	165	19.88	119	408.70	226	467.80
5	0.5	120028	5	17648	15.32	17546	426.06	17741	392.83
			8	9290	19.01	9059	568.31	9142	511.19
			10	6516	34.55	6413	818.39	6692	621.76
5	0.9	215751	5	37064	13.51	37180	339.82	37137	468.41
			8	21341	18.84	2166	828.66	21353	416.56
			10	16162	19.66	16170	545.96	16258	534.37

REFERENCES

1. Chams, M., Hertz, A. and de Werra, D. "Some experiments with simulated annealing for coloring graphs," *European Journal of Operational Research* 32 (1987), pp. 260-266.
2. de Werra, D., "An introduction to timetabling, "*European Journal of Operational Research*" 3 (1985), pp. 151-162.
3. Hertz, A., "A Fast Algorithm For Coloring Meyniel Graphs," *Journal of combinatorial Theory, Series B* 50 (1990), pp. 231-240.
4. Hertz, A., "Tabu Search For Large Scale Timetabling Problem," *European Journal of Operational Research* 54 (1991), p. 39-47.
5. Hertz, A., "Cosine : A new graph coloring algorithm," *Operations Research Letters* 10 (1991), pp. 411-415.
6. J. R. Brown, "Chromatic Scheduling And The Chromatic Number Problem," *Management Science* 19 (1972), pp. 456-463.
7. Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. "Optimization by simulated annealing : an experimental evaluation ; part Ⅱ, graph coloring and number partitioning," *Operations Researh* 39 (1991), pp. 378-406.
8. Kernighan, B. and Lin, S., "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal* 49 (1970), pp. 291-307.
9. Krajewski, L. J. and Ritzman, L. P. *Operations Management*, Addison-Wesley Publishing Company Inc., 1990.
10. Leighton, F. T. "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards* 84 (1979), pp. 489-506.
11. Park, T. "Algorithms for Partitioning a graph," To appear in *Computers and Industrial Engineering*.