Optimization by Simulated Catalytic Reaction: Application to Graph Bisection

Yong-Hyuk Kim[1] and Seok-Joong Kang[2*]
[1] School of Software, Kwangwoon University
Seoul, 01897 – South Korea
[e-mail: yhdfly@kw.ac.kr]
[2] Department of Management of Technology for Defense, Korea University
Seoul, 02841 – South Korea
[e-mail: sjkang64@korea.ac.kr]
*Corresponding author: Seok-Joong Kang

Received October 27, 2017; revised December 3, 2017; accepted December 22, 2017; published May 31, 2018

Abstract

Chemical reactions have an intricate relationship with the search for better-quality neighborhood solutions to optimization problems. A catalytic reaction for chemical reactions provides a clue and a framework to solve complicated optimization problems. The application of a catalytic reaction reveals new information hidden in the optimization problem and provides a non-intuitive perspective. This paper proposes a new simulated catalytic reaction method for search in optimization problems. In the experiments using this method, significantly improved results are obtained in almost all graphs tested by applying to a graph bisection problem, which is a representative problem of combinatorial optimization problems.

Keywords: Search, optimization, simulated catalytic reaction, graph bisection

The present research has been conducted by the Research Grant of Kwangwoon University in 2017. This research was supported by a grant [KCG-01-2017-05] through the Disaster and Safety Management Institute funded by Korea Coast Guard of Korean government, and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2015R1D1A1A01060105).

http://doi.org/10.3837/tiis.2018.05.014 **ISSN : 1976-7277**

1. Introduction

Optimization is one of the major fields of computer science, and many studies have been conducted to obtain realistic solutions to optimization problems, which for a long time have been generally difficult to solve. Natural world phenomena have been successfully imitated for use in some specific computing applications. For example, simulated annealing (SA) [1] and large-step Markov chain (LSMC) [2] methods that mimic annealing in physics and evolutionary computation (e.g., genetic algorithms [3]) that imitate biological evolution have been successfully implemented over the past 30 years and are still being actively researched. These methods are all iterative search methods that operate by improving the state of the solution, and they have the unique ability to search for attractive solution spaces that have never been found before. Rather than attempting to directly address the limitations of the current state, these methods are based on probabilistic techniques to search for a new area.

When using iterative search methods, it is easy to fall into a local optimum that is better than neighboring solutions rather than moving states, and the ability to overcome this problem determines the performance of the search algorithm. Once falling into the local optimum, it is not easy to move to a better state because the ‘cluster’ becomes the 'barrier' of the state. Resolving cluster obstacles requires a fairly large 'energy' to naturally overcome the barrier.

Overcoming the barrier and shifting the state from the local optimum to a better solution is similar to overcoming energy thresholds in chemical reactions. If the energy does not increase above the threshold, the chemical reaction never occurs. By lowering the threshold using a catalyst, it is easier to cause a reaction. This study proposes a search method that simulates a catalytic reaction to implement a search technique mimicking a catalytic reaction. By lowering the energy threshold, it is possible to naturally move between clusters. The proposed method is applied to a graph bisection problem, which is a representative problem of combinatorial optimizations, and the results were greatly improved.

The contributions of this paper are as follows:

- Proposal of a novel search method that mimics a catalytic reaction.
- Application of the proposed method to the graph bisection problem and deriving favorable results.

The remainder of this paper is as follows. Section 2 presents the simulated catalytic reaction method proposed in this paper. Section 3 introduces the graph bisection problem, which is a test problem for the proposed method, and describes our experiments. Finally, the conclusion is discussed in Section 4.

2. Simulated Catalytic Reaction

Fig. 1 shows a common hill-climbing state search process. From the initial state, the method moves to a better neighboring state, and this process is repeated until there is no further improvement. If there is no better neighbor, the search is terminated, and this final solution becomes a local optimum. SA [1], which mimics its namesake phenomenon in physics, is a technique that opens the possibility of moving to a worse neighbor by using the concept of temperature. This section proposes a new technique to overcome the local optimum by moving to a better state. This technique is designed by mimicking chemical reactions.

```
Let s ← a random initial state;
do
        Pick a random neighbor of s, s_new ← neighbor(s);
        if s is better than s_new, then s ← s_new;
until (there is no improvement)
return the final state s;
```

Fig. 1. Common state search process (hill climbing)

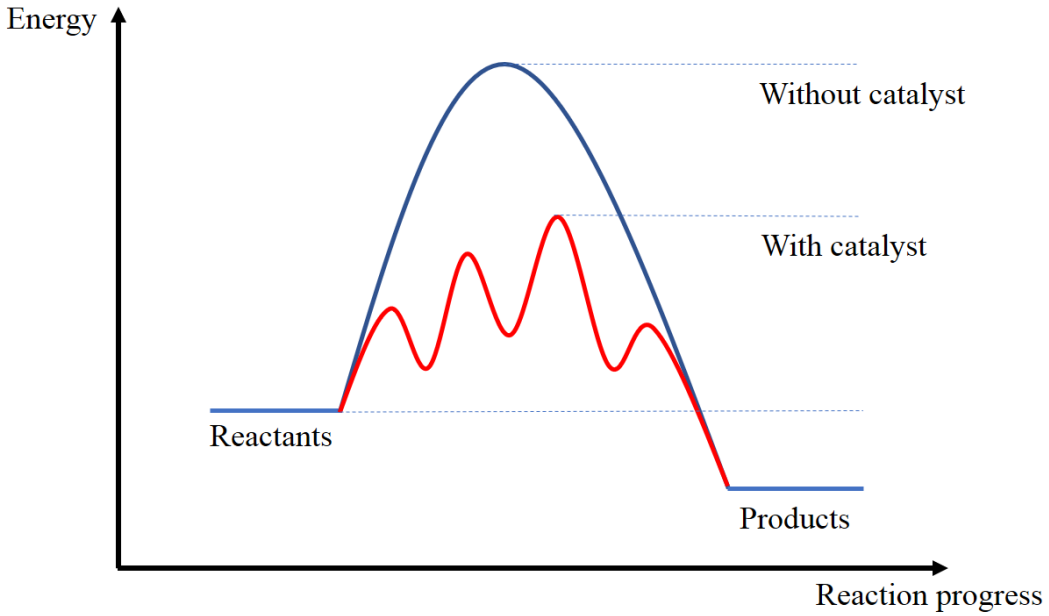


Fig. 2. Catalytic reaction in chemistry

Fig. 2 shows the catalytic reaction process in chemistry. Reactants use a catalyst to lower the energy threshold in situations where much energy is needed to make products, and the reaction can occur more easily when the energy threshold is lowered. **Fig. 3** shows a simulated catalytic reaction process. To overcome the local optimum and advance to a better solution, the 'energy threshold' is lowered using a catalyst-like relaxed gain. The energy threshold in the search corresponds to the minimization objective. Even if the quality of the solution deteriorates, it must be overcome to achieve a global optimum. By changing the state transition in the gain-based method by relaxed gain, it increases the possibility of movement to the global optimum.

Fig. 4 shows the search method combined with the simulated catalytic reaction proposed in this paper. The search is divided into three stages. The first step is to find a 'cluster' that corresponds to the barrier of the search from the local optima, which is difficult to improve in

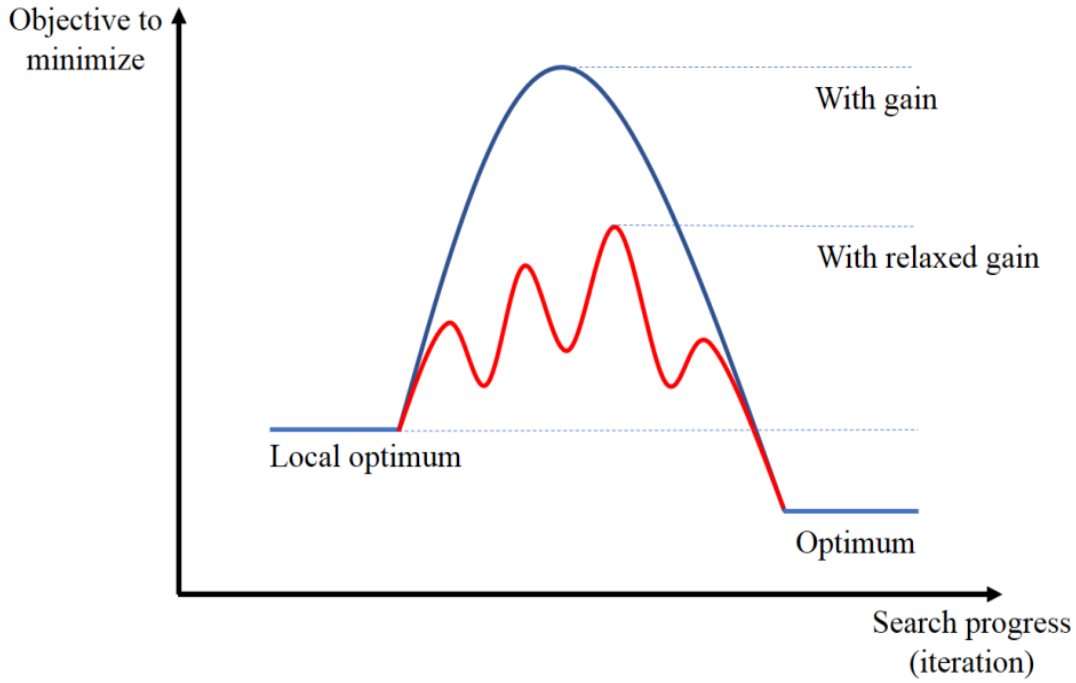


Fig. 3. Simulated catalytic reaction in search

a conventional way. This step can be specified only when the problem to be solved is determined, and various methods can be applied. The second step weakens the connectivity in the cluster to move clusters, and then it uses the relaxed gain based on this to lower the 'energy threshold' to perform the search. The design of the relaxed gain that weakens the connectivity in the cluster can be specified only when the problem that is to be solved has been determined. The final step is shown in **Fig. 1**, which stabilizes the unstable reaction. If more searches are necessary, then the last two steps can be repeated.

Let $s \leftarrow$ a random initial state;
Find barriers from local optima; // *step of finding barriers*
do // *step of simulated catalytic reaction*
    Pick a random neighbor of s, $s_{new} \leftarrow$ neighbor(s);
    if s is better than s_{new} in a distorted space with relaxed gain, **then** $s \leftarrow s_{new}$;
until (there is no improvement)
do // *step of general hill climbing*
    Pick a random neighbor of s, $s_{new} \leftarrow$ neighbor(s);
    if s is better than s_{new} in the original space, **then** $s \leftarrow s_{new}$;
until (there is no improvement)
return the final state s;

Fig. 4. Search with simulated catalytic reaction

3. Combinatorial Optimization and Test Problem

3.1 Combinatorial Optimization

Most research in combinatorial optimization is aimed at efficiently finding the maximum or minimum value of a function with many independent variables. This function is usually called a cost function or an objective function and is a quantitative measure of the "goodness" of a complex system. The cost function relates to the detailed configuration of the system and is related to optimization problems that arise from the physical design of the particular computer system. However, most combinatorial optimization problems that are of real interest belong to the NP-complete problem [4], which means that it is quite difficult to obtain the optimal solution within a limited time. So far, interest has focused on designing heuristic methods with a low computational complexity.

Heuristic methods generally yield reasonable solutions within a realistic time frame, but this method is problem-specific and cannot guarantee the optimal solution. The next section discusses the graph bisection problem, which is a representative combinatorial, NP-complete optimization problem (i.e., NP-hard problem). This problem is also an optimization problem that occurs in VLSI circuit design.

3.2 Graph Bisection and Testbed

Let $G = (V, E)$ be an unweighted, undirected graph, where V is a set of nodes and E is a set of edges. The bisection $\{V_1, V_2\}$ of graph G satisfies the following conditions: V_1 and V_2 are subsets of V with no overlap, and the union of the two sets is equal to the whole set V. Also, the difference in cardinality between the two sets is less than or equal to one. The cut size of the bisection $\{V_1, V_2\}$ is defined by the number of edges that span the two sets. The problem of finding a bisection with a minimum cut size is called the graph bisection problem. Formally, the cut size of the bisection $\{V_1, V_2\}$ is defined as follows: $|\{(v, w) \in E: v \in V_1 \text{ and } w \in V_2\}|$, where $V_1 \subset V$, $V_2 \subset V$, $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, and $\big||V_1| - |V_2|\big| = 1$. This problem is NP-hard for general graphs [4], meaning that the polynomial time algorithm that solves this problem is not present in current technology. In other words, finding the optimal solution of the graph bisection problem is a very difficult task. At present, designing approximate or heuristic algorithms that solve the problem is the only solution. There are also various meta-heuristic approaches to solving the graph bisection, including genetic algorithms [5,6,7] and tabu search [8,9]. Kim et al. [10] present a survey of techniques using genetic algorithms.

When two bisections are different in one or two nodes, they are referred to as neighbors. For each bisection, there are $O(|V|^2)$ neighbors associated with a single node move. The gain of a node is calculated as the amount reduction in the cut size obtained when it moves from the current node subset to the opposite. In a pure random walk, all connections have the same transition probability. However, in the gain-based search algorithm, only transitions with a positive gain value have non-zero transition probabilities. In a typical graph, this algorithm can never reach the optimal solution. In this paper, to overcome this disadvantage by changing the transition probability by momentarily relaxing the gain value. Random perturbation could be a simple solution to this problem, but it is difficult to present a "correct direction". The rational and natural directions for perturbation are presented in the following sections.

The experiments are conducted on well-known benchmark graphs used in numerous studies [5,6,8,11,12,13,14,15,16,17]. The proposed algorithm is applied to 28 well-known graphs [6]. The number of nodes in the graphs tested ranged from 350 to 5,200. All benchmark graphs are available at http://soar.snu.ac.kr/benchmark. A detailed description of the graph is shown

below.

- U*n.d*: a random geometric graph with *n* nodes and an average degree of each node to be *d*. The nodes are arbitrarily placed in the unit rectangle, and an edge is present between nodes having an Euclidean distance of t or less. The value of *d* is πnt^2.
- breg.*n.d*: a random regular graph with *n* nodes and the degree of each node to be 3. Optimal bisection cut size is *b*.
- cat.*n*: a caterpillar graph with *n* nodes. Nodes are placed in spline, and each node has 6 legs. The bisection cut size of all used caterpillar graphs is 1.
- w-grid.*n.d*: Grid graph with donut shape without boundary with *n* nodes. The optimal bisection cut size is *b*.

3.3 Clustering Method

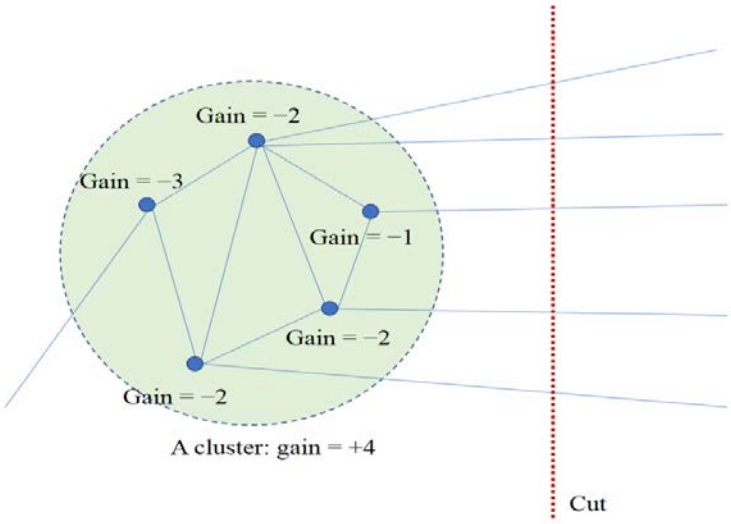
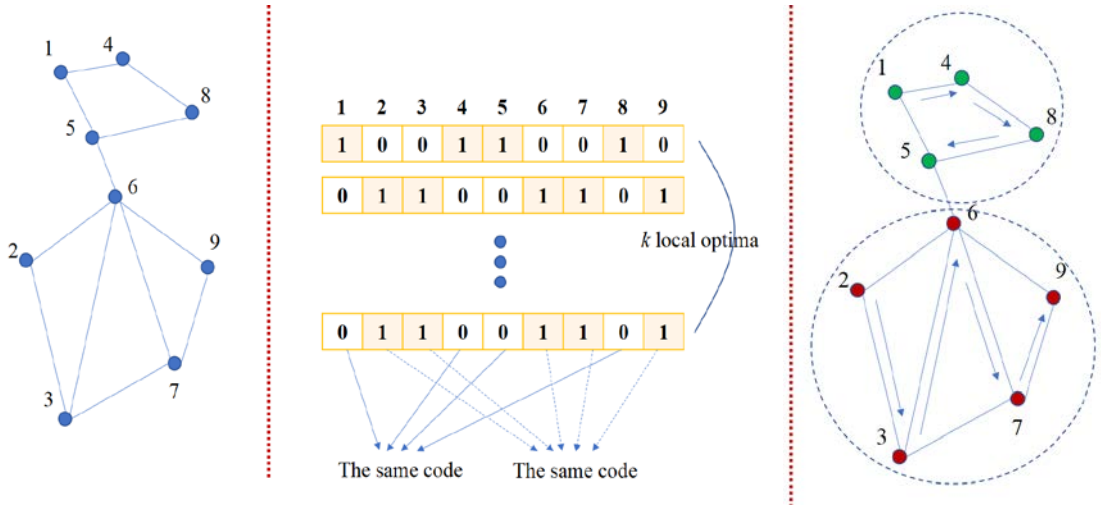


Fig. 5. An example of a cluster

In general, a cluster is a subgraph with a high density of linkages. **Fig. 5** shows an example of a 5-node cluster. Let us define the gain of the node or cluster as the reduction in cut size when the node or cluster moves to the opposite side. Then, the gain of the cluster in **Fig. 5** is +4, but the gain of all nodes in the cluster is negative. This example shows that it is very difficult to move a cluster by moving one node at a time.

Graph clustering is used to reduce the search space. For example, clustering could improve the Fiduccia-Mattheyses (FM) algorithm [18] using a two-step methodology [19,20]. The clustering method used in this paper is shown in **Figs. 6** and **7**. This paper does not focus on suggesting a clustering method with superior performance. Hwang and Kim [7] and Yoon and Kim [21] described recent clustering methodologies. Instead, the simple clustering method proposed by Kim [22] is used in this paper. In this method, the cluster is defined as a strongly correlated subgraph with respect to local optimization. In other words, all nodes in the cluster have almost always belonged to the same partition in the local optima. One solution to the graph bisection problem is expressed as $|V|$ bit code. Each bit corresponds to a node in the graph. If a node belongs to V_1 in the partition, it has a value of 0, and if it belongs to V_2, it has a value of 1.



(a) G=(V,E) (b) Finding local optima (c) Clustering with (b)

Fig. 6. Example process of the used clustering

Clustering proceeds as follows: First, k local optima are collected. Then, $k|V|$ bit code is obtained, and each node has k-bit binary string information. If the k-bit binary strings of the two nodes are the same, there is strong evidence that a cluster is present, and the nodes are considered to be included in the cluster. The clustering method used searches for nodes with the same k-bit binary string using a depth-first search to extract the node subset. **Fig. 6** shows an example of the execution result of the clustering used. Also, **Fig. 7** shows the pseudo-code

Get m local optima $l1, l2, \ldots, lm$,
where $lk = (lk_1, lk_2, \ldots, lk_{|V|})$ for each $k = 1, 2, \ldots, m$;
Get a string code $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{|V|})$ by setting
$\alpha_i \leftarrow (l1_i, l2_i, \ldots, lm_i)$ for each $i = 1, 2, \ldots, |V|$;
$W \leftarrow V$;
do
Choose a vertex v in W;
Find a cluster C including v through a depth-first search,
i.e., $C \leftarrow$ the set of vertices in W with the same binary string as α_v in α;
if $|C| > 1$, **then** $W \leftarrow W \backslash C$;
else $W \leftarrow W \backslash \{v\}$;
until (W is empty)

Fig. 7. The clustering method used [22]

of the clustering method used. In this experiment, k was 50. Each node can be selected exactly once. Assuming that the maximum node degree is constant, the time complexity of the clustering method used is $O(|V|)$. Two-stage FM with this clustering improved the FM

significantly [22], indicating that it effectively detected the cluster.

Table 1 shows the cluster information obtained from the graph used in this paper. There are differences in the size and number of clusters according to the graph. Except for one graph (w-grid100.20), the average cluster size ranges from 2 to 500. Depending on the cluster size, the search performance will also be affected.

Table 1. Cluster information of tested graphs

Graph name	#nodes	Ave. cluster size	#clusters
U500.05	500	5.82	79
U500.10	500	8.91	53
U500.20	500	11.12	41
U500.40	500	45.27	11
U1000.05	1000	6.30	149
U1000.10	1000	9.17	102
U1000.20	1000	12.21	77
U1000.40	1000	20.29	48
U2000.05	2000	6.07	311
U2000.10	2000	8.89	213
U5000.05	5000	6.31	752
U5000.10	5000	10.03	474
breg500.0	500	45.45	11
breg500.12	500	9.00	49
breg500.16	500	9.62	45
breg500.20	500	6.59	64
breg5000.0	5000	500.00	10
breg5000.4	5000	81.80	61
breg5000.8	5000	94.02	53
breg5000.16	5000	56.48	88
cat.352	352	4.80	50
cat.702	702	4.73	100
cat.1052	1052	5.01	150
cat.5252	5252	5.99	750
w-grid100.20	100	N.A.	0
w-grid500.42	500	2.08	61
w-grid1000.40	1000	7.73	127
w-grid5000.100	5000	10.97	453

3.4 Barrier Avoidance

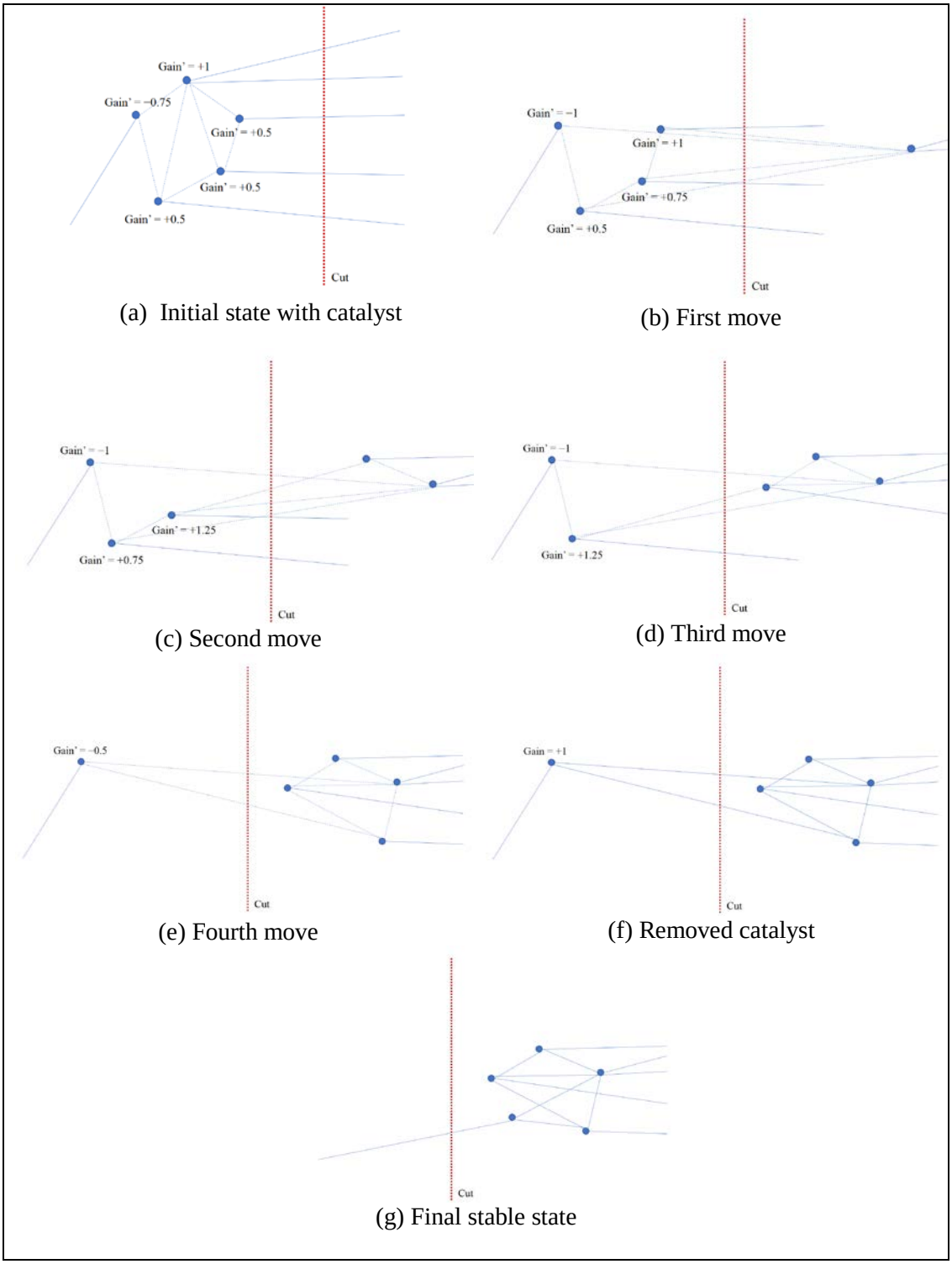


Fig. 8. Cluster moving by simulated catalytic reaction

If a group of nodes has a strong connection to one another and consequently has a strong pattern in the local optima, moving one node at a time will make it very difficult to break the pattern. This is due to the strong nature of the pattern not to be broken, and it requires a very large perturbation ("energy") to break it. The clusters mentioned in the previous section have this kind of pattern, and moving an entire cluster directly seems to be able to solve the problem of falling into local optima, but this is a very unnatural way. This approach does not only result in a lack of confidence in the pattern, but it also does not take into account the constraint of balancing the two sets of nodes that make up the bisection. In this study, a new method is used to relax the gain instantly by weakening the connection in the group of nodes found in the cluster and breaks the pattern naturally. (Weakened to half the level in our actual experiments). It is a way to overcome the barrier naturally, not too far from the "right direction" of the search.

When the SCR of FM was applied to the graph bisection problem, the cost function of the bisection $\{V_1, V_2\}$ is formally defined as follows: $|\{(v,w) \in E: v \in V_1,\ w \in V_2, \text{and } \{v,w\} \nsubseteq C\}| + \rho|\{(v,w) \in E: v \in V_1,\ w \in V_2, \text{and } \{v,w\} \subseteq C\}|$, where $C \subseteq V$ is a cluster and $0 \leq \rho < 1$. **Fig. 8** shows an example that the cluster composed of 5 nodes in **Fig. 5** can be moved to the opposite side by a simulated catalytic reaction. The connectivity in the cluster has been changed to 1/4 level using the virtual catalyst (**Fig. 8(a)**). As a result, most of the relaxed gain values of the nodes in the cluster are positive (+0.5 or higher, except for the leftmost node). This reduces the energy required to move the cluster, and allows the nodes in the cluster to move one-by-one to the opposite side (**Figs. 8(b) - (e)**). The last one node in the cluster cannot move because the relaxed gain is negative (**Fig. 8(e)**), but it will move when the original gain is restored (**Fig. 8(f)**). Eventually the entire cluster can move. (**Fig. 8(g)**).

do
  Compute gain gv for each $v \in V$;
  Make gain lists of g_vs;
  $M \leftarrow$ empty set;
  for $i \leftarrow 1$ **to** $|V| - 1$
    Choose $v_i \in V - M$ such that g_{vi} is maximal and
      the move of vi does not violate the balance criterion;
    $M \leftarrow M \cup \{v_i\}$;
    for each $v \in V - M$ adjacent to v_i
      Update its gain g_v and adjust the gain list;
  Choose $k \in \{1, 2, \ldots, |V| - 1\}$ that maximizes $\Sigma\, i{=}1..k\ g_{vi}$;
  Move all the vertices in the subset $\{v_1, v_2, \ldots, v_k\}$ to their opposite sides;
until (there is no improvement)

Fig. 9. Pseudo-code of the Fiduccia-Mattheyses algorithm [18]

3.5 Experiments

3.5.1 Experimental Setting and Basic Search

The experiment was conducted on the 28 benchmark graphs mentioned in Section 3.2. The

CPU used was an Intel (R) Core (TM) i5 CPU 760, 2.80GHz, and the testing program was written in the *C* language. The Fiduccia-Mattheyses (FM) algorithm was used as a basic search algorithm and was improved by using the simulated catalytic reaction method presented in Section 2. The FM algorithm is a representative iterative improvement algorithm, and further details are as follows.

For nearly 35 years, the Kernighan-Lin (KL) algorithm [23] has been used as a benchmark algorithm for graph partitioning. Fiduccia and Matteyses [18] proposed a linear time algorithm that improved the KL algorithm. They were able to significantly reduce the time per pass while pursuing a bisection with some imbalance. The FM algorithm as well as the KL algorithm are traditional iterative improvement algorithms. The FM algorithm improves the initial solution by moving the nodes one by one. The KL algorithm, on the other hand, improves the solution by swapping the node pairs. **Fig. 9** shows the pseudo-code of the FM algorithm. The FM algorithm performs a series of passes. It moves every node in each pass, and the best solution during the present pass becomes the initial solution of the next pass. This algorithm terminates

Table 2. Experimental results of the FM algorithm and its corresponding SCR search

Graph name	FM algorithm [18] Best/Ave/SD[1]	Time[2] (ms)	SCR of FM Best/Ave/SD[1]	Time[2] (ms)	*t*-test *p*-value
U500.05	8/38.33/9.77	0.3	**5**/**24.73**/8.61	0.6	1.3e-162
U500.10	26/90.72/27.70	0.4	26/**56.90**/20.24	1.1	6.6e-150
U500.20	178/223.09/39.57	0.7	178/**198.95**/20.15	1.8	1.6e-58
U500.40	412/435.32/56.58	1.0	412/**416.31**/19.32	2.6	5.0e-23
U1000.05	26/74.81/15.68	0.7	**15**/**47.73**/13.03	1.5	2.5e-223
U1000.10	55/160.56/46.27	1.2	**39**/**98.41**/32.50	2.7	1.7e-174
U1000.20	222/316.42/69.69	1.9	222/**274.49**/40.80	4.4	4.0e-54
U1000.40	737/860.68/132.33	2.6	737/**788.50**/84.09	6.8	5.5e-44
U2000.05	82/161.37/24.95	1.5	**39**/**102.97**/19.56	3.3	0.0e-∞
U2000.10	119/344.12/73.78	2.7	**72**/**228.34**/61.12	6.0	5.2e-198
U5000.05	252/438.52/47.74	5.1	**151**/**280.28**/33.47	11.2	0.0e-∞
U5000.10	563/956.93/133.08	8.3	**237**/**605.39**/116.46	19.1	0.0e-∞
breg500.0	0/14.92/34.57	0.2	0/**4.48**/15.16	0.4	4.6e-18
breg500.12	12/48.61/22.49	0.2	12/**40.27**/21.55	0.6	4.4e-17
breg500.16	16/55.24/21.91	0.2	16/**44.53**/19.45	0.6	2.1e-29
breg500.20	20/69.27/18.55	0.2	20/**67.62**/18.75	0.5	2.4e-2
breg5000.0	0/15.43/121.64	3.4	0/**5.33**/70.00	4.6	1.2e-2
breg5000.4	4/71.93/162.14	4.2	4/**13.55**/43.33	7.2	6.0e-27
breg5000.8	8/118.44/169.09	4.8	8/**25.92**/62.67	6.9	5.0e-53
breg5000.16	16/172.25/165.16	6.8	16/**55.95**/68.42	10.5	5.3e-79
cat.352	7/21.33/5.54	0.1	**5**/**19.32**/5.72	0.1	2.0e-15
cat.702	19/41.34/9.27	0.1	**13**/**36.22**/9.12	0.2	1.7e-33
cat.1052	29/59.11/12.53	0.2	**25**/**49.49**/10.83	0.4	1.9e-65
cat.5252	177/253.03/32.31	1.3	**126**/**197.98**/33.09	2.7	3.9e-194
w-grid100.20	20/21.89/2.97	0.0	20/21.97/3.03	0.1	2.8e-1
w-grid500.42	42/48.77/8.82	0.2	42/**46.67**/6.45	0.4	8.7e-10
w-grid1000.40	40/47.46/13.98	0.5	40/**45.31**/11.37	1.2	8.5e-05
w-grid5000.100	100/122.43/36.86	4.7	100/**108.48**/17.80	11.0	5.3e-26

[1] Results from 1,000 independent runs.
[2] CPU milliseconds on an Intel Core i5 760, 2.80GHz.

when no better solution is found. When combined with an efficient data structure called a bucket list, the time complexity of the FM algorithm is $O(|E|)$. The FM algorithm can be viewed as a variant k-opt scheme, in which the number of nodes moved per pass is not fixed, and this improves the one-opt scheme in which only one node is changed per pass.

Table 3. Experimental results of the KL algorithm and the SCR of FM

Graph name	KL algorithm [23] Best/Ave/SD[1]	Time[2] (ms)	SCR of FM Best/Ave/SD[1]	Time[2] (ms)	*t*-test *p*-value
U500.05	12/38.29/9.90	0.4	5/24.73/8.61	0.6	2.8e-160
U500.10	26/90.20/26.88	0.7	26/56.90/20.24	1.1	9.6e-151
U500.20	178/221.01/40.31	1.1	178/198.95/20.15	1.8	6.5e-49
U500.40	412/436.57/45.71	1.6	412/416.31/19.32	2.6	1.1e-35
U1000.05	26/74.57/16.11	1.1	15/47.73/13.03	1.5	2.0e-216
U1000.10	51/163.84/45.44	1.9	39/98.41/32.50	2.7	4.9e-190
U1000.20	222/312.13/69.82	3.0	222/274.49/40.80	4.4	7.8e-45
U1000.40	737/860.00/142.31	4.3	737/788.50/84.09	6.8	1.8e-39
U2000.05	75/162.35/25.47	2.7	39/102.97/19.56	3.3	0.0e-∞
U2000.10	71/346.93/77.93	4.9	72/228.34/61.12	6.0	1.2e-195
U5000.05	251/433.93/47.16	13.0	151/280.28/33.47	11.2	0.0e-∞
U5000.10	469/956.87/139.32	21.3	237/605.39/116.46	19.1	0.0e-∞
breg500.0	0/16.48/23.54	0.3	0/4.48/15.16	0.4	7.7e-39
breg500.12	12/49.82/14.46	0.4	12/40.27/21.55	0.6	9.6e-30
breg500.16	16/55.12/13.66	0.5	16/44.53/19.45	0.6	1.5e-41
breg500.20	20/69.91/15.06	0.5	20/67.62/18.75	0.5	1.3e-3
breg5000.0	0/23.49/25.50	22.9	0/5.33/70.00	4.6	1.5e-14
breg5000.4	4/65.45/57.17	24.1	4/13.55/43.33	7.2	8.3e-94
breg5000.8	8/115.08/48.73	26.2	8/25.92/62.67	6.9	1.1e-179
breg5000.16	16/188.29/35.17	32.7	16/55.95/68.42	10.5	2.3e-301
cat.352	7/20.70/4.91	0.1	5/19.32/5.72	0.1	4.7e-9
cat.702	19/41.80/7.91	0.2	13/36.22/9.12	0.2	2.7e-44
cat.1052	29/58.36/10.07	0.3	25/49.49/10.83	0.4	4.5e-69
cat.5252	177/251.79/28.53	6.1	126/197.98/33.09	2.7	6.1e-203
w-grid100.20	20/21.91/2.32	0.0	20/21.97/3.03	0.1	3.1e-1
w-grid500.42	42/47.84/7.57	0.4	42/46.67/6.45	0.4	1.1e-4
w-grid1000.40	40/47.45/13.54	1.0	40/45.31/11.37	1.2	6.9e-05
w-grid5000.100	100/122.39/38.98	23.9	100/108.48/17.80	11.0	7.1e-24

[1] Results from 1,000 independent runs.
[2] CPU milliseconds on Intel Core i5 760, 2.80 GHz

3.5.2 Results

The FM algorithm was used as the basic search method, and it improved through the SCR method proposed in this paper. **Table 2** shows the experimental results based on 1,000 times of independent runs. The best results, average results, and standard deviation of each algorithm are also presented in **Table 2**. The probability p of the one-tailed t-test is presented in the last column of the table to statistically verify the performance difference between the two algorithms. The smaller the p value is, the more reliable the performance difference is. At a 95% confidence level (i.e., p-value less than 0.05), there were better quality improvements in all test graphs except for one graph (w-grid100.20).

This one graph is a graph in which no clusters are found, as can be seen in **Table 1**, naturally, no SCR effect can be expected. Each algorithm was performed in less than 20ms in all cases, and the results were obtained very quickly. The FM algorithm is a result of convergence until there are no further improvements, so even with more time, quality cannot be improved.

The SCR of FM was compared with the KL algorithm [23], which is another representative search method in graph bisection. The results are shown in **Table 3**. It could be seen that the results were consistent with those of comparing with the FM algorithm in **Table 2**.

4. Conclusion

This paper proposes a simulated catalytic reaction method mimicking the catalytic reaction in the field of chemistry. As far as the authors know, this is the first trial to apply the phenomenon of chemistry to a computation. The proposed method can be successfully applied to graph bisection, which is a typical problem in combinatorial optimization field.

In this study, a very simple clustering method [22] was used to find clusters that require large energy for movement. However, an improvement in its performance can be expected when more advanced clustering methods [7,21,24,25,26,27,28,29,30,31] are applied.

In addition, this study is applied to a graph bisection problem that is representative of combinatorial optimization, but it can also be applied to general combinatorial optimization problems as well, and application to other problems is left for future research. There are two issues that need to be discussed to apply the SCR to other problems: (i) How to define 'cluster'? This issue is closely related to the encoding of the solution. A ‘cluster’ to apply SCR to a problem is related to local optimizer of the problem. A `cluster’ in this study is a barrier of a basis search algorithm, and then it can be defined as the *unbreakable pattern* in the encodings of local optima with respect to the algorithm. From this definition, SCR can overcome the defect of a given local optimizer. (ii) How to weaken connectivity within the cluster. The issue is closely related to the cost function of the problem. When the proposed SCR was applied to another combinatorial optimization problem in which the cost function of a solution S is $\sum_{u:\ \text{partial encoding}} \text{cost}(S_u)$, the transformed cost function becomes $\sum_{u \not\subseteq C} \text{cost}(S_u) + \rho \sum_{u \subseteq C} \text{cost}(S_u)$, where C is a cluster and $0 \leq \rho < 1$.

References

[1] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. “Optimization by simulated annealing,” *Science 220(4598)*, pp. 671–680, 1983. Article (CrossRef Link)

[2] Hong, I., Kahng, A.B., Moon, B.-R. “Improved Large-Step Markov Chain Variants for the Symmetric TSP,” *Journal of Heuristics*, vol. 3, no. 1, pp. 63–81, 1997. Article (CrossRef Link)

[3] Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, 1989.

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

[5] Bui, T.N., Moon, B.-R. “Genetic algorithm and graph partitioning,” *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 841–855, 1996. Article (CrossRef Link)

[6] Kim, Y.-H., Moon, B.-R. “Lock-gain based graph partitioning,” *Journal of Heuristics*, vol. 10, no. 1, pp. 37–57, 2004. Article (CrossRef Link)

[7] Hwang, I., Kim, Y.-H., Yoon, Y. “Moving clusters within a memetic algorithm for graph partitioning,” *Mathematical Problems in Engineering*, Vol. 2015, Article ID 238529, 2015. Article (CrossRef Link)

[8] Battiti, R., Bertossi, A. "Greedy, prohibition, and reactive heuristics for graph partitioning," *IEEE Transactions on Computers*, vol. 48, no. 4, pp. 361–385, 1999. Article (CrossRef Link)
[9] Kalayci, T.E., Battiti, R. "A reactive self-tuning scheme for multilevel graph partitioning," *Applied Mathematics and Computation*, vol. 318, pp. 227–244, 2018. Article (CrossRef Link)
[10] Kim, J., Hwang, I., Kim, Y.-H., Moon, B.-R. "Genetic approaches for graph partitioning: a survey," in *Proc. of of the Genetic and Evolutionary Computation Conference*, pp. 473-480, 2011. Article (CrossRef Link)
[11] Hwang, I., Kim, Y.-H., Moon, B.-R. "Multi-attractor gene reordering for graph bisection," in *Proc. of the Eighth Annual Conference on Genetic and Evolutionary Computation*, pp. 1209–1216, 2006. Article (CrossRef Link)
[12] Johnson, D.S., Aragon, C., McGeoch, L., Schevon, C. "Optimization by simulated annealing: An experimental evaluation, Part 1, graph partitioning," *Operations Research*, vol. 37, pp. 865–892, 1989. Article (CrossRef Link)
[13] Kim, Y.-H., Moon, B.-R. "Investigation of the fitness landscapes in graph bipartitioning: An empirical study," *Journal of Heuristics*, vol. 10, no. 2, pp. 111–133, 2004. Article (CrossRef Link)
[14] Merz, P., Freisleben, B. "Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning," *Evolutionary Computation*, vol. 8, no. 1, pp. 61–91, 2000. Article (CrossRef Link)
[15] Moraglio, A., Kim, Y.-H., Yoon, Y., Moon, B.-R. "Geometric crossovers for multiway graph partitioning," *Evolutionary Computation*, vol. 15, no. 4, pp. 445–474, 2007. Article (CrossRef Link)
[16] Yoon, Y., Kim, Y.-H. "New bucket managements in iterative improvement partitioning algorithms," *Applied Mathematics & Information Sciences*, vol. 7, no. 2, pp. 529-532, 2013. Article (CrossRef Link)
[17] Seo, K., Hyun, S., Kim, Y.-H. "An edge-set representation based on spanning tree for searching cut space," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 465-473, 2015. Article (CrossRef Link)
[18] Fiduccia, C., Mattheyses, R. "A linear time heuristic for improving network partitions," in *Proc. of Proceedings of the 19th ACM/IEEE Design Automation Conference*, pp. 175–181, 1982. Article (CrossRef Link)
[19] Alpert, C., Kahng, A.B. "A general framework for vertex orderings, with applications to netlist clustering," in *Proc. of Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 63–67, 1994. Article (CrossRef Link)
[20] Bui, T.N., Heigham, C., Jones, C., Leighton, T. "Improving the performance of the Kernighan-Lin and simulated annealing graph bisection algorithms," in *Proc. of Proceedings of the 26th ACM/IEEE Design Automation Conference*, pp. 775–778, 1989. Article (CrossRef Link)
[21] Yoon, Y., Kim, Y.-H. "Vertex ordering, clustering, and their application to graph partitioning," *Applied Mathematics & Information Sciences*, vol. 8, no. 1, pp. 135-138, 2014. Article (CrossRef Link)
[22] Kim, Y.-H. "An enzyme-inspired approach to surmount barriers in graph bisection," in *Proc. of Proceedings of the International Conference on Computational Science and Its Applications - Lecture Notes in Computer Science 5072*, pp. 841-851, 2008. Article (CrossRef Link)
[23] Kernighan, B., Lin, S. "An efficient heuristic procedure for partitioning graphs," *Bell Systems Technical Journal 49*, pp. 291–307, 1970. Article (CrossRef Link)
[24] Alpert, C.J., Kahng, A.B. "Recent directions in netlist partitioning: a survey," *Integration, the VLSI Journal*, vol. 19, no. 1-2, pp. 1–81, 1995. Article (CrossRef Link)
[25] Choe, T.-Y., Park, C.-I. "A *k*-way graph partitioning algorithm based on clustering by eigenvector," in *Proc. of Proceedings of the Fourth International Conference on Computational Science*, pp. 598–601, 2004. Article (CrossRef Link)
[26] Cong, J., Lim, S.K. "Edge separability-based circuit clustering with application to multilevel circuit partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 346–357, 2004. Article (CrossRef Link)

[27] Dhillon, I., Guan, Y., Kulis, B. "A fast kernel-based multilevel algorithm for graph clustering," in *Proc. of Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge discovery in data mining*, pp. 629–634, 2005. Article (CrossRef Link)
[28] Huang, M.L., Nguyen, Q.V. "A fast algorithm for balanced graph clustering" in *Proc. of Proceedings of the Eleventh International Conference on Information Visualization*, pp. 46–52, 2007. Article (CrossRef Link)
[29] Saha, B., Mitra, P. "Dynamic algorithm for graph clustering using minimum cut tree," in *Proc. of Proceedings of the Sixth IEEE International Conference on Data Mining Workshops*, pp. 667–671, 2006. Article (CrossRef Link)
[30] Schaeffer, S.E. "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007. Article (CrossRef Link)
[31] Wang, J., Peng, H., Hu, J., Yang, C. "A graph clustering algorithm based on minimum and normalized cut," in *Proc. of Proceedings of the Seventh International Conference on Computational Science*, pp. 497–504, 2007. Article (CrossRef Link)

Yong-Hyuk Kim received the B.S. degree in computer science, and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University, Seoul, Korea, in 1999, 2001, and 2005, respectively. Since 2007 he has been a Professor with the Department of Computer Science and Engineering, Kwangwoon University, Seoul. His research interests include algorithm design/analysis, discrete mathematics, optimization theory, combinatorial optimization, evolutionary computation, operations research, data/web mining, machine learning, smart grids, and sensor networks. Dr. Kim has served as an Editor of the KSII Transactions on Internet and Information Systems from 2010 to 2013. He is a member of the IEEE SMC Society and the IEEE Communications Society.

Seok-Joong Kang received his BS and MS degrees in Computer Science from Indiana University in 1988 and 1991 respectively and his Ph.D degree in Electrical Engineering and Computer Science from University of California, Irvine in 2003. After the completion of his Ph.D program, he worked as a lecturer and research staff at UCI. During 1991-1998, he worked as a senior researcher at Korea Institute of Defense Analyses in Republic of Korea. He also worked as a principle researcher at Samsung Electronics Co. during 2004-2006. Currently, he is an associate professor in the department of Management of Technology for Defense at Korea University, Seoul, Korea. His research interest includes Embedded System, Software engineering, Optimization, Real-time system and Distributed systems.