Solving the Bi-criteria Max-Cut Problem with Different Neighborhood Combination Strategies

Li-Yuan Xue[1], Rong-Qiang Zeng[2,3(✉)], Zheng-Yin Hu[3], and Yi Wen[3]

[1] EHF Key Laboratory of Science, School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu 611731, Sichuan, People's Republic of China
xuely2013@gmail.com

[2] School of Mathematics, Southwest Jiaotong University, Chengdu 610031, Sichuan, People's Republic of China
zrq@swjtu.edu.cn

[3] Chengdu Documentation and Information Center, Chinese Academy of Sciences, Chengdu 610041, Sichuan, People's Republic of China
{huzy,wenyi}@clas.ac.cn

Abstract. Local search is known to be a highly effective metaheuristic framework for solving a number of classical combinatorial optimization problems, which strongly depends on the characteristics of neighborhood structure. In this paper, we integrate different neighborhood combination strategies into the hypervolume-based multi-objective local search algorithm, in order to deal with the bi-criteria max-cut problem. The experimental results indicate that certain combinations are superior to others and the performance analysis sheds lights on the ways to further improvements.

Keywords: Multi-objective optimization · Hypervolume contribution · Neighborhood combination · Local search · Max-cut problem

1 Introduction

Local search is a simple and effective metaheuristic framework for solving a number of classical combinatorial optimization problems, which proceeds from an initial solution with a sequence of local changes by defining the proper neighborhood structure for the considered problem. In order to study different neighborhood combination strategies during the local search process, we present the experimental analysis of different neighborhoods to solve the bi-criteria max-cut problem.

Given an undirected graph $G = (V, E)$ with the vertex set $V = \{1, \ldots, n\}$ and the edge set $E \subset V \times V$. Each edge $(i, j) \in E$ is associated with a weight w_{ij}. The max-cut problem is to seek a partition of the vertex set V into two disjoint subsets V_1 and V_2, which is mathematically formulated as follows [1]:

$$f_k(V_1, V_2) = max \sum_{i \in V_1, j \in V_2} w_{ij}^k \tag{1}$$

H. Yin et al. (Eds.): IDEAL 2017, LNCS 10585, pp. 508–515, 2017.
https://doi.org/10.1007/978-3-319-68935-7_55

where w_{ij}^k is the weight of the k^{th} ($k \in \{1, 2\}$) graph. As one of Karps 21 NP-complete problems with numerous practical applications [4], a large number of metaheuristics have been proposed to tackle this problem, including scatter search [6], global equilibrium search [7], tabu search [8], etc.

In this paper, we integrate different neighborhood combination strategies into the hypervolume-based multi-objective local search algorithm, in order to study the search capability of different neighborhood combinations on the bi-criteria max-cut problem. The experimental results indicate that certain combinations are superior to others. The performance analysis explains the behavior of the algorithms and sheds lights on the ways to further enhance the search.

The remaining part of this paper is organized as follows. In the next section, we briefly introduce the basic notations and definitions of bi-objective optimization. In Sect. 3, we present the hypervolume-based multi-objective local search algorithm with different neighborhood combination strategies for solving bi-criteria max-cut problem. Section 4 indicates that the experimental results on the benchmark instances of max-cut problem. The conclusions are provided in the last section.

2 Bi-objective Optimization

In this section, we briefly introduce the basic notations and definitions of bi-objective optimization. Without loss of generality, we assume that X denotes the search space of the optimization problem under consideration and $Z = \Re^2$ denotes the corresponding objective space with a maximizing vector function $Z = f(X)$, which defines the evaluation of a solution $x \in X$ [5]. Specifically, the dominance relations between two solutions x_1 and x_2 are presented below [9]:

Definition 1 *(Pareto Dominance). A decision vector x_1 is said to dominate another decision vector x_2 (written as $x_1 \succ x_2$), if $f_i(x_1) \geq f_i(x_2)$ for all $i \in \{1, 2\}$ and $f_j(x_1) > f_j(x_2)$ for at least one $j \in \{1, 2\}$.*

Definition 2 *(Pareto Optimal Solution). $x \in X$ is said to be Pareto optimal if and only if there does not exist another solution $x' \in X$ such that $x' \succ x$.*

Definition 3 *(Non-Dominated Solution). $x \in S$ ($S \subset X$) is said to be non-dominated if and only if there does not exist another solution $x' \in S$ such that $x' \succ x$.*

Definition 4 *(Pareto Optimal Set). S is said to be a Pareto optimal set if and only if S is composed of all the Pareto optimal solutions.*

Definition 5 *(Non-dominated Set). S is said to be a non-dominated set if and only if any two solutions $x_1 \in S$ and $x_2 \in S$ such that $x_1 \nsucc x_2$ and $x_2 \nsucc x_1$.*

Actually, we are interested in finding the Pareto optimal set, which keeps the best compromise among all the objectives. However, it is very difficult or even impossible to generate the Pareto optimal set in a reasonable time for the NP-hard problems. Therefore, we aim to obtain a non-dominated set which is as close to the Pareto optimal set as possible. That's to say, the whole goal is to identify a Pareto approximation set with high quality.

3 Neighborhood Combination Strategies

In this work, we integrate different neighborhood combination strategies into the hypervolume-based multi-objective local search algorithm, in order to solve bi-criteria max-cut problem. The general scheme of Hypervolume-Based Multi-Objective Local Search (HBMOLS) algorithm [3] is presented in Algorithm 1, and the main components of this algorithm are described in detail in the following subsections.

Algorithm 1. Hypervolume-Based Multi-Objective Local Search Algorithm

Input: N (Population size)
Output: A: (Pareto approximation set)
Step 1 - Initialization: $P \leftarrow N$ randomly generated solutions
Step 2: $A \leftarrow \Phi$
Step 3 - Fitness Assignment: Assign a fitness value to each solution $x \in P$
Step 4:
while Running time is not reached **do**
  repeat
    Hypervolume-Based Local Search: $x \in P$
  until all neighbors of $x \in P$ are explored
    $A \leftarrow$ Non-dominated solutions of $A \bigcup P$
end while
Step 5: Return A

In HBMOLS, each individual in an initial population is generated by randomly assigning the vertices of the graph to the two vertex subsets V_1 and V_2. Then, we employ a Hypervolume Contribution (HC) indicator proposed in [3] to achieve the fitness assignment for each individual. Based on the dominance relation and two objective function values, the HC indicator calculate the hypervolume contribution of each individual in the objective space.

For the hypervolume-based local search procedure, we implement the f-flip ($f \in \{1, 2\}$) move based neighborhood strategy with the combinations. Afterwards, each solution is optimized by the hypervolume-based local search procedure, which will repeat until the termination criterion is satisfied, so as to obtain a high-quality Pareto approximation set.

3.1 One-Flip Move

In order to deal with the max-cut problem, one-flip move is realized by moving a randomly selected vertex to the opposite set, which is calculated as follows:

$$\Delta_i = \sum_{x \in V_1, x \neq v_1} w_{v_i x} - \sum_{y \in V_2} w_{v_i y}, \;\; v_i \in V_1 \tag{2}$$

$$\Delta_i = \sum_{x \in V_2, y \neq v_1} w_{v_i y} - \sum_{y \in V_1} w_{v_i x}, \;\; v_i \in V_2 \tag{3}$$

Let Δ_i be the move gain of representing the change in the fitness function, and Δ_i can be calculated in linear time by the formula above, more details about this formula can be found in [8]. Then, we can calculate the objective function values high efficiently with the streamlined incremental technique.

3.2 Two-Flip Move

In the case of two-flip move, we can obtain a new neighbor solution by randomly moving two different vertices v_i and v_j from the set V_1 to another set V_2. In fact, two-flip move can be seen as a combination of two single one-flip moves. We denote the move value by δ_{ij}, which is derived from two one-flip moves Δ_i and Δ_j $(i \neq j)$ as follows:

$$\delta_{ij} = \Delta_i + \Delta_j \tag{4}$$

Especially, the search space generated by two-flip move is much bigger than the one generated by one-flip move. In the following, we denote the neighborhoods with one-flip move and two-flip move as N_1 and N_2 respectively.

4 Experiments

In this section, we present the experimental results of 3 different neighborhood combination strategies on 9 groups of benchmark instances of max-cut problem. All the algorithms are programmed in C++ and compiled using Dev-C++ 5.0 compiler on a PC running Windows 7 with Core 2.50 GHz CPU and 4 GB RAM.

4.1 Parameters Settings

In order to conduct the experiments on the bi-objective max-cut problem, we use two single-objective benchmark instances of max-cut problem with the same dimension provided in [4][1] to generate one bi-objective max-cut problem instance. All the instances used for experiments are presented in Table 1 below.

In addition, the algorithms need to set a few parameters, we only discuss two important ones: the running time and the population size, more details about the parameter settings for multi-objective optimization algorithms can be found in [2,8]. The exact information about the parameter settings in our work is presented in the following Table 2.

4.2 Performance Assessment Protocol

In this paper, we evaluate the efficacy of 3 different neighborhood combination strategies with the performance assessment package provided by Zitzler et al.[2].

[1] More information about the benchmark instances of max-cut problem can be found on this website: http://www.stanford.edu/~yyye/yyye/Gset/.

[2] More information about the performance assessment package can be found on this website: http://www.tik.ee.ethz.ch/pisa/assessment.html.

Table 1. Single-objective benchmark instances of max-cut problem used for generating bi-objective max-cut problem instances.

	Dimension	Instance 1	Instance 2
bo_mcp_800_01	800	g1.rud	g2.rud
bo_mcp_800_02	800	g11.rud	g12.rud
bo_mcp_800_03	800	g15.rud	g19.rud
bo_mcp_800_04	800	g17.rud	g21.rud
bo_mcp_2000_01	2000	g22.rud	g23.rud
bo_mcp_2000_02	2000	g32.rud	g33.rud
bo_mcp_2000_03	2000	g35.rud	g39.rud
bo_mcp_1000_01	1000	g43.rud	g44.rud
bo_mcp_3000_01	3000	g49.rud	g50.rud

Table 2. Parameter settings used for bi-objective max-cut problem instances: instance dimension (D), vertices (V), edges(E), population size (P) and running time (T).

	Dimension (D)	Vertices (V)	Edges (E)	Population (P)	Time (T)
bo_mcp_800_01	800	800	19176	20	$40''$
bo_mcp_800_02	800	800	1600	20	$40''$
bo_mcp_800_03	800	800	4661	20	$40''$
bo_mcp_800_04	800	800	4667	20	$40''$
bo_mcp_2000_01	2000	2000	19990	50	$100''$
bo_mcp_2000_02	2000	2000	4000	50	$100''$
bo_mcp_2000_03	2000	2000	11778	50	$100''$
bo_mcp_1000_01	1000	1000	9990	25	$50''$
bo_mcp_3000_01	3000	3000	6000	75	$150''$

The quality assessment protocol works as follows: First, we create a set of 20 runs with different initial populations for each strategy and each benchmark instance of max-cut problem. Then, we generate the reference set RS^* based on the 60 different sets $A_0, \ldots, A_{59}$ of non-dominated solutions.

According to two objective function values, we define a reference point $z = [r_1, r_2]$, where r_1 and r_2 represent the worst values for each objective function in the reference set RS^*. Afterwards, we assign a fitness value to each non-dominated set A_i by calculating the hypervolume difference between A_i and RS^*. Actually, this hypervolume difference between these two sets should be as close to zero as possible [10].

4.3 Computational Results

In this subsection, we present the computational results on 9 groups of bi-objective max-cut problem instances, which are obtained by three different neighborhood combination strategies. The information about these algorithms are described in the following table:

Table 3. The algorithms with different neighborhood combination strategies.

	Algorithm description
HBMOLS_N_1	One-flip move based local search
HBMOLS_N_2	Two-flip move based local search
HBMOLS_$(N_1 \bigcup N_2)$	f-flip move based local search ($f \in \{1, 2\}$)

In Table 3, the algorithms HBMOLS_$(N_1 \bigcup N_2)$ selects one of the two neighborhoods to be implemented at each iteration during the local search process, choosing the neighborhood N_1 with a predefined probability p and choosing N_2 with the probability $1 - p$. In our experiments, we set the probability $p = 0.5$.

The computational results are summarized in Table 4. In this table, there is a value both **in bold** and **in grey box** at each line, which is the best result obtained on the considered instance. The values both **in italic** and **bold** at each line refer to the corresponding algorithms which are **not** statistically outperformed by the algorithm obtaining the best result (with a confidence level greater than 95%).

Table 4. The computational results on bi-criteria max-cut problem obtained by the algorithms with 4 different neighborhood combination strategies.

Instance	Algorithms		
	N_2	$N_1 \bigcup N_2$	N_1
bo_mcp_800_01	0.176056	0.131597	**0.115592**
bo_mcp_800_02	0.165386	0.138711	**0.104922**
bo_mcp_800_03	0.151565	0.147590	**0.117074**
bo_mcp_800_04	0.198746	***0.142533***	**0.134102**
bo_mcp_2000_01	0.243824	0.231131	**0.176594**
bo_mcp_2000_02	0.152572	***0.098441***	**0.090894**
bo_mcp_2000_03	0.377184	0.352036	**0.304157**
bo_mcp_1000_01	0.279807	0.260862	**0.229748**
bo_mcp_3000_01	0.099988	0.098586	**0.092150**

From Table 4, we can observe that all the best results are obtained by N_1, which statistically outperforms the other two algorithms on all the instances. Moreover, the results obtain by $N_1 \bigcup N_2$ is close to the results obtained by N_1. Especially, the most significant result is achieved on the instance bo_mcp_2000_01, where the average hypervolume difference value obtained by N_1 is much smaller than the values obtained by the other two algorithms.

Nevertheless, N_2 does not perform as well as N_1, although the search space of them is much bigger than N_1. We suppose that there exists some key vertices in the representation of the individuals, which means these vertices should be assigned in some set in order to search the local optima effectively. Two-flip moves change the positions of the key vertices much more frequently than the one-flip move, then the efficiency of local search is obviously affected by this neighborhood strategy. Actually, $N_1 \bigcup N_2$ provides a possibility to keep the positions of the key vertices unchanged and broaden the search space. Thus, the combination of one-flip move and two-flip move is very potential to obtain better results.

5 Conclusion

In this paper, we have presented different neighborhood combination strategies to deal with the bi-criteria max-cut problem, which are based on one-flip, two-flip and the combination. For this purpose, we have carried out the experiments on 9 groups of benchmark instances of max-cut problem. The experimental results indicate that the better outcomes are achieved with the simple one-flip move based neighborhood and the neighborhood combination with two-flip is very potential to escape the local optima for further improvements.

Acknowledgments. The work in this paper was supported by the Fundamental Research Funds for the Central Universities (Grant No. A0920502051722-53) and supported by the West Light Foundation of Chinese Academy of Science (Grant No: Y4C0011001).

References

1. Angel, E., Gourves, E.: Approximation algorithms for the bi-criteria weighted max-cut problem. Discrete Appl. Math. **154**, 1685–1692 (2006)
2. Basseur, M., Liefooghe, A., Le, K., Burke, E.: The efficiency of indicator-based local search for multi-objective combinatorial optimisation problems. J. Heuristics **18**(2), 263–296 (2012)
3. Basseur, M., Zeng, R.-Q., Hao, J.-K.: Hypervolume-based multi-objective local search. Neural Comput. Appl. **21**(8), 1917–1929 (2012)
4. Benlic, U., Hao, J.-K.: Breakout local search for the max-cut problem. Eng. Appl. Artif. Intell. **26**, 1162–1173 (2013)
5. Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer, Secaucus (2007)

6. Marti, R., Duarte, A., Laguna, M.: Advanced scatter search for the max-cut problem. INFORMS J. Comput. **21**(1), 26–38 (2009)
7. Shylo, V.P., Shylo, O.V.: Solving the maxcut problem by the global equilibrium search. Cybern. Syst. Anal. **46**(5), 744–754 (2010)
8. Wu, Q., Wang, Y., Lü, Z.: A tabu search based hybrid evolutionary algorithm for the max-cut problem. Appl. Soft Comput. **34**, 827–837 (2015)
9. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30217-9_84
10. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. Evol. Comput. **3**, 257–271 (1999)