Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 475018, 16 pages
doi:10.1155/2012/475018

Research Article

A VNS Metaheuristic with Stochastic Steps for Max 3-Cut and Max 3-Section

Ai-fan Ling[1,2]

[1] *Research Center of Security and Future, School of Finance, Jiangxi University of Finance and Economics, Nanchang 330013, China*
[2] *Key Laboratory of Management, Decision and Information Systems, Academy of Mathematics and Systems Science, CAS, Beijing 100190, China*

Correspondence should be addressed to Ai-fan Ling, aifanling@yahoo.com.cn

Received 15 February 2012; Accepted 30 May 2012

Academic Editor: John Gunnar Carlsson

A heuristic algorithm based on VNS is proposed to solve the Max 3-cut and Max 3-section problems. By establishing a neighborhood structure of the Max 3-cut problem, we propose a local search algorithm and a variable neighborhood global search algorithm with two stochastic search steps to obtain the global solution. We give some numerical results and comparisons with the well-known 0.836-approximate algorithm. Numerical results show that the proposed heuristic algorithm can obtain efficiently the high-quality solutions and has the better numerical performance than the 0.836-approximate algorithm for the NP-Hard Max 3-cut and Max 3-section problems.

1. Introduction

Given a graph $G(V;E)$, with nodes set $V$and edges set E, the Max 3-cut problem is to find a partition $S_0 \subset V$, $S_1 \subset V$ and $S_2 \subset V$, of the set V, such that $S_0 \bigcup S_1 \bigcup S_2 = V$, $S_i \bigcap S_j = \emptyset (i \neq j)$ and the sum of the weights on the edges connecting the different parts is maximized. Similar to the Max cut problem, the Max 3-cut problem has long been known to be NP complete [1], even for any un-weighted graphs [2], and has also applications in circuit layout design, statistical physics, and so on [3]. However, due to the complexity of this problem, its research progresses is much lower than that of the Max cut problem. Based on the semidefinite programming relaxation proposed by Goemans and Williamson [4], Frieze and Jerrum [5] obtained a 0.800217-approximation algorithm for the Max 3-Cut problem. Recently, Goemans and Williamson [6] and Zhang and Huang [7] improved Frieze and

Jerrum's 0.800217-approximation ratio to 0.836 using a complex semidefinite programming relaxation of the Max 3-cut problem.

For the purpose of our analysis, we first introduce some notations. We denote the complex conjugate of $y = a + ib$ by $\overline{y} = a - ib$, where $i = \sqrt{-1}$ is the pure image number and the real part and image part of a complex number by $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$, respectively. For an n dimensional complex vector $\mathbf{y} \in \mathbb{C}^n$ written as bold letter and n dimensional complex matrix $Y \in \mathbb{C}^{n\times n}$, we write $\mathbf{y}^*$ and Y^* to denote their conjugate and transpose. That is, $\mathbf{y}^* = \overline{\mathbf{y}}^T$ and $Y^* = \overline{Y}^T$. The set of n dimensional real symmetric (semidefinite positive) matrices and the set of n dimensional complex Hermitian (semidefinite positive) matrices are denoted by $\mathcal{S}_n(\mathcal{S}_n^+)$ and $\mathcal{H}_n(\mathcal{H}_n^+)$, respectively. We sometimes use $A \succcurlyeq 0$ to show $A \in \mathcal{S}_n^+$(or $A \in \mathcal{H}_n^+$). For any two complex vector $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$, we write $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^*\mathbf{v}$ as their inner product. For any two complex matrices $A, B \in \mathcal{H}_n$, we write $\langle A, B \rangle = A \cdot B$ as their inner product; that is, $\langle A, B \rangle = A \cdot B = \mathrm{Tr}(B^*A) = \sum_{i,j} \overline{b}_{ij} a_{ij}$, where $A = (a_{ij})$ and $B = (b_{ij})$. $\|\cdot\|$ means the module of a complex number or the 2-norm of a complex vector or the F-norm of a complex matrix.

Let the third root of unity be denoted by $\omega^0 = 1$, $\omega = \omega^1 = e^{i(2\pi/3)}$, $\omega^2 = e^{i(4\pi/3)}$. Introduce a complex variable $y_i \in \{1, \omega, \omega^2\}$, $i = 1, \ldots, n$, then it is not hard to know that

$$\frac{2}{3} - \frac{1}{3} y_i \cdot y_j - \frac{1}{3} y_j \cdot y_i = \frac{2}{3}\left(1 - \mathrm{Re}\left(y_i \cdot y_j\right)\right). \tag{1.1}$$

Denote $S_k = \{i : y_i = \omega^k, k = 0, 1, 2\}$ and $\mathbf{y} = (y_1, \ldots, y_n)^T$. Then the Max 3-cut problem can be expressed as

$$\begin{aligned} \text{M3C}: \ \max f(\mathbf{y}) &= \frac{2}{3} \sum_{i<j} w_{ij}\left(1 - \mathrm{Re}\left(y_i \cdot y_j\right)\right) \\ \text{s.t.} \quad & \mathbf{y} \in \{1, \omega, \omega^2\}^n, \end{aligned} \tag{1.2}$$

here $\mathbf{y} \in \{1, \omega, \omega^2\}^n$ means that $y_i \in \{1, \omega, \omega^2\}$, $i = 1, \ldots, n$, $W = (w_{ij})$ is the weight-valued matrix of a given graph.

By relaxing the complex variable y_i into an n dimensional complex vector $\mathbf{y}_i$, we get a complex semidefinite programming (CSDP) relaxation of (M3C) as follows:

$$\begin{aligned} \text{CSDP}: \ \max & \frac{2}{3} \sum_{i<j} w_{ij}\left(1 - \mathrm{Re}\left(\mathbf{y}_i \cdot \mathbf{y}_j\right)\right) \\ \text{s.t.} \quad & \|\mathbf{y}_i\| = 1, \quad i = 1, 2, \ldots, n, \\ & A_{ij}^k \cdot Y \geq -1, \quad i, j = 1, 2, \ldots, n, \ k = 0, 1, 2 \\ & Y \succcurlyeq 0, \end{aligned} \tag{1.3}$$

where $Y_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$, $A_{ij}^k = \omega^k \mathbf{e}_i \mathbf{e}_j^T + \omega^{-k} \mathbf{e}_j \mathbf{e}_i^T$ and $\mathbf{e}_i$ denotes the vector with zeros everywhere except for an unit in the ith component. It is easily to verify that constraints $A_{ij}^k \cdot Y \geq -1$ can be expressed as

$$\mathrm{Re}(\omega^k Y_{ij}) \geq -\frac{1}{2}, \quad k = 0, 1, 2. \tag{1.4}$$

To get an approximate solution of M3C, Goemans and Williamson [6] do not directly solve the CSDP, but solve an equivalent real SDP with following form (Although some softwares, such as SeDuMi [8] and the earlier version of SDPT3-4.0 [9], can deal with SDPs with complex data, this does not reduce the dimensions of problems):

$$\begin{aligned}
\text{RSDP}: \max\ & \frac{1}{2}\begin{bmatrix} Q & O \\ O & Q \end{bmatrix} \cdot X \\
\text{s.t.}\ & \begin{bmatrix} \mathbf{e}_i \mathbf{e}_i^T & O \\ O & \mathbf{e}_i \mathbf{e}_i^T \end{bmatrix} \cdot X = 2, \quad i = 1, 2, \ldots, n, \\
& \begin{bmatrix} \mathrm{Re}\left(A_{ij}^k\right) & -\mathrm{Im}\left(A_{ij}^k\right) \\ \mathrm{Im}\left(A_{ij}^k\right) & \mathrm{Re}\left(A_{ij}^k\right) \end{bmatrix} \cdot X \geq -2, \quad 1 \leq i < j \leq n,\ k = 0, 1, 2 \\
& \begin{bmatrix} A_{ij}^0 & O \\ O & -A_{ij}^0 \end{bmatrix} \cdot X = 0, \quad 1 \leq i < j \leq n, \\
& \begin{bmatrix} O & A_{ij}^0 \\ A_{ij}^0 & O \end{bmatrix} \cdot X = 0, \quad 1 \leq i < j \leq n, \\
& \begin{bmatrix} O & \mathbf{e}_i \mathbf{e}_i^T \\ \mathbf{e}_i \mathbf{e}_i^T & O \end{bmatrix} \cdot X = 0, \quad i = 1, 2, \ldots, n, \\
& X \in \mathcal{S}_+^{2n},
\end{aligned} \tag{1.5}$$

where $Q = (1/3)\,\mathrm{diag}(We) - W$ is the Laplace matrix of given graph, O is an n-dimensional full zeros matrix.

In RSDP, the first, third, and forth classes of equality constraints ensure that $X_{ii} = 1$, $i = 1, 2, \ldots, n$ and with the form

$$X = \begin{bmatrix} R & -S \\ S & R \end{bmatrix}. \tag{1.6}$$

The final two classes of equality constraints ensure that $S_{ii} = 0$ $(i = 1, \ldots, n)$ and S is a skew-symmetric matrix.

If X is an optimal solution of RSDP, then the complex matrix $\widehat{Y} = R + iS$ is an optimal solution of CSDP. Then one can randomly generate a complex vector ξ, such that $\xi \sim N(0, \widehat{Y})$, and set

$$\widehat{y}_i = \begin{cases} 1, & \text{if } \operatorname{Arg}(\xi_i) \in \left[0, \dfrac{2\pi}{3}\right), \\ \omega, & \text{if } \operatorname{Arg}(\xi_i) \in \left[\dfrac{2\pi}{3}, \dfrac{4\pi}{3}\right), \\ \omega^2, & \text{if } \operatorname{Arg}(\xi_i) \in \left[\dfrac{4\pi}{3}, 2\pi\right), \end{cases} \tag{1.7}$$

where $\operatorname{Arg}(\cdot) \in [0, 2\pi]$ means the complex angle principal value of a complex number. Goemans and Williamson [6] had verified that, see also Zhang and Huang [7],

$$f(\widehat{\mathbf{y}}) \geq 0.836 \cdot \left(Q \cdot \widehat{Y}\right). \tag{1.8}$$

The algorithm proposed by Goemans and Williamson [6] can obtain a very good approximate ratio, and RSDP can be solved by interior point algorithm, but the 0.836-approximate algorithm will be not practical in numerical study for the Max 3-cut problem. From RSDP, one can see that for a graph with n nodes, RSDP has $2n + 5n(n-1)/2$ constraints and $3n(n-1)/2$ slack variables via the inequality constraints. That is to say, RSDP has a $2n$ dimensional unknown symmetrical semidefinite positive matrix variable and a $3n(n-1)/2$ dimensional unknown vector variable, and $2n + 5n(n-1)/2$ constraints, and has also many matrices without an explicit block diagonal structure although they are sparse. For instance, when $n = 100$, RSDP becomes a very-high-dimensional semidefinite programming problem with 14850 slack variables and 24950 constraints. Further, as we known, it is only a class of universal and medium-scale instances for Max 3-cut problems with 50 to 100 nodes. Hence, it will be very time consuming to solve such a RSDP relaxation of M3C using the current existing any SDP softwares. This leads that 0.836-approximate algorithm is not suitable for computational study of the Max 3-cut problem. This limitation for solving M3C based on CSDP (or RSDP) relaxation motivates us to find a new efficient and fast algorithm for the practical purpose for the Max 3-cut problem.

In the current paper, we first establish a definition of K-neighborhood structure of the Max 3-cut problem and design a local search algorithm to find the local minimizer. And then, we propose a variable neighborhood search (VNS) metaheuristic with stochastic steps which is originally considered by Mladenović and Hansen [10], by which we can find efficiently a high-quality global approximate solution of the Max 3-cut problem. Further, combining a greed algorithm, we extend the proposed algorithm to the Max 3-section problem. To the best of our knowledge, it is first time to consider the computational study of the Max 3-cut problem. In order to test the performance of the proposed algorithm, we compare the numerical results with Goemans and Williamson's 0.836-approximate algorithm.

This paper is organized as follows. In Section 2, we give some definitions and lemmas. In Section 3, we present the VNS metaheuristic for solving the Max 3-cut problem. The VNS is extended to the Max 3-section problem in Section 4. In Section 5, we give some numerical results and comparisons.

2. Preliminaries

In this section, we will establish some definitions and give some facts for our sequel purpose. For the third roots of unity, $1, \omega, \omega^2$, we can get the following fact:

$$\|1-\omega\|^2 = \left\|\omega-\omega^2\right\|^2 = \left\|1-\omega^2\right\|^2 = 3. \tag{2.1}$$

Denote $\mathbb{S} = \{1, \omega, \omega^2\}^n$. Then based on (2.1), for any $\mathbf{y} \in \mathbb{S}$, we may definite a K-neighborhood of $\mathbf{y}$ as follows.

Definition 2.1. For any $\mathbf{y} \in \mathbb{S}$ and any positive integer number K $(1 \le K \le n)$, one defines the K-neighborhood, denoted by $N_K(\mathbf{y})$, of $\mathbf{y}$ as the set

$$N_K(\mathbf{y}) = \left\{\mathbf{z} \in \mathbb{S} : \|\mathbf{z}-\mathbf{y}\|^2 = \sum_{i=1}^{n}\left\|z_i - y_i\right\|^2 \le 3K\right\}. \tag{2.2}$$

In particular, if $K = 1$, we write the 1-neighborhood $N_1(\mathbf{y})$ of $\mathbf{y}$ as $N(\mathbf{y})$.

The boundary of the K-neighborhood $N_K(\mathbf{y})$ is defined by $\partial N_K(\mathbf{y}) = \{\mathbf{z} \in \mathbb{S} : \|\mathbf{y}-\mathbf{z}\|^2 = 3K\}$. Clearly, $N(\mathbf{y}) = \partial N(\mathbf{y})$. If $\mathbf{z} \in \partial N_K(\mathbf{y})$, we call $\mathbf{z}$ a K-neighbor of $\mathbf{y}$. From Definition 2.1, the difference of between points $\mathbf{y}$ and its K-neighbor $\mathbf{z}$ is that they have only K different components. By computing straightforwardly, we get the number of elements of $\partial N_K(\mathbf{y})$, that is $|\partial N_K(\mathbf{y})| = 2^K C_n^K$. Particularly, $|\partial N(\mathbf{y})| = 2n$ when $K = 1$.

Example 2.2. Let $\mathbf{y} = (\omega, \omega, \omega^2)^T \in \{1, \omega, \omega^2\}^3$. Then $(1, \omega, \omega^2)^T \in N(\mathbf{y})$, $(1, \omega^2, \omega^2)^T \in \partial N_2(\mathbf{y}) \subset N_2(\mathbf{y})$, and $(1, \omega^2, \omega)^T \in \partial N_3(\mathbf{y}) \subset N_3(\mathbf{y})$.

Definition 2.3. For any $u \in \{0, 1, 2\}$, define two maps from $\{1, \omega, \omega^2\}$ to itself as follows: $\tau_i(\omega^u) = \omega^{u+i} \in \{1, \omega, \omega^2\}, i = 1, 2$.

Clearly, for any $u \in \{0, 1, 2\}$, $\tau_i(\omega^u) \ne \omega^u$, $i = 1, 2$ and $\tau_1(\omega^u) \ne \tau_2(\omega^u)$. Applying Definition 2.3, for any $\mathbf{z} \in N(\mathbf{y})$ there exists an unique component, z_k say, of $\mathbf{z}$, such that $z_k \ne y_k$ and either $z_k = \tau_1(y_k)$ or $z_k = \tau_2(y_k)$, and other components of $\mathbf{z}$ and $\mathbf{y}$ are the same. For simplicity, for any $\mathbf{z} \in N(\mathbf{y})$ with $z_k \ne y_k$ and $z_i = y_i\,(i = 1, \ldots, n, i \ne k)$, we denote by $\mathbf{z} = \tau_1^k(\mathbf{y})$ or $\mathbf{z} = \tau_2^k(\mathbf{y})$ corresponding to $z_k = \tau_1(y_k)$ or $z_k = \tau_2(y_k)$. By Definitions 2.1 and 2.3, for any $\mathbf{y} \in \mathbb{S}$, we can structure its 1-neighborhood points using maps defined by Definition 2.3; that is, we have the following result.

Lemma 2.4. *Let $\tau_i(\cdot)$ $(i = 1, 2)$ be defined by Definition 2.3. Then, for any $\mathbf{y} \in \mathbb{S}$ and any fixed positive integer number k $(1 \le k \le n)$, one has*

$$\tau_i^k(\mathbf{y}) \in N(\mathbf{y}), \quad i = 1, 2, \tag{2.3}$$

that is, $\tau_1^k(\mathbf{y})$ and $\tau_2^k(\mathbf{y})$ are two 1-neighborhood points of $\mathbf{y}$.

Definition 2.5. A point $\hat{\mathbf{y}} \in \mathbb{S}$ is called a K-local maximizer of the function f over $\mathbb{S}$, if $f(\hat{\mathbf{y}}) \ge f(\mathbf{y})$, for all $\mathbf{y} \in N_K(\hat{\mathbf{y}})$. Furthermore, if $f(\hat{\mathbf{y}}) \ge f(\mathbf{y})$ for all $\mathbf{y} \in \mathbb{S}$, then $\hat{\mathbf{y}}$ is called a global

maximizer of f over $\mathbb{S}$. A 1-local maximizer of the function f is also called a local maximizer of the function f over $\mathbb{S}$.

3. VNS for Max 3-Cut

3.1. Local Search Algorithm

Let $\mathbf{y}^0 = (y_1^0, \ldots, y_n^0)^T \in \mathbb{S}$ be a feasible solution of problem M3C. If $\mathbf{y}^0$ is not a local maximizer of f, then for all $\mathbf{y} \in N(\mathbf{y}^0)$, we may find a $\tilde{\mathbf{y}} \in N(\mathbf{y}^0)$, such that $f(\tilde{\mathbf{y}}) = \max\{f(\mathbf{y}) : \mathbf{y} \in N(\mathbf{y}^0)\}$. It is clear that $f(\tilde{\mathbf{y}}) \geq f(\mathbf{y}^0)$. If $\tilde{\mathbf{y}}$ is not still a local maximizer of f, then replacing $\mathbf{y}^0$ with $\tilde{\mathbf{y}}$ and repeating the process until a point $\hat{\mathbf{y}}$ satisfying $f(\hat{\mathbf{y}}) = \max\{f(\mathbf{y}) : \mathbf{y} \in N(\hat{\mathbf{y}})\}$ is found, which indicates that $\hat{\mathbf{y}}$ is a local maximizer of f.

For any positive integer number k $(1 \leq k \leq n)$, let $\mathbf{y}^k = (y_1^k, \ldots, y_n^k)^T = \tau_i^k(\mathbf{y}^0) \in N(\mathbf{y}^0)$ $(i = 1, 2)$; that is,

$$\begin{gathered} y_i^k = y_i^0, \quad i = 1, 2, \ldots, k-1, k+1, \ldots, n; \\ y_k^k \neq y_k^0. \end{gathered} \tag{3.1}$$

Denote

$$\delta(k) = f\left(\mathbf{y}^0\right) - f\left(\mathbf{y}^k\right). \tag{3.2}$$

Then, we have the following result whose proof is clear.

Lemma 3.1. *Consider*

$$\delta(k) = \begin{cases} \dfrac{2}{3}\displaystyle\sum_{i=1}^{k-1} w_{ik} \operatorname{Re}\left[y_i^0 \cdot \left(y_k^k - y_k^0\right)\right] + \dfrac{2}{3}\sum_{j=k+1}^{n} w_{kj} \operatorname{Re}\left[\left(y_k^k - y_k^0\right) \cdot y_j^0\right], & k > 1; \\ \dfrac{2}{3}\displaystyle\sum_{j=k+1}^{n} w_{kj} \operatorname{Re}\left[\left(y_k^k - y_k^0\right) \cdot y_j^0\right], & k = 1. \end{cases} \tag{3.3}$$

Based on Lemma 3.1, if we know the value of $f(\mathbf{y}^0)$, then we can obtain the value function $f(\mathbf{y}^k)$ at next iterative point $\mathbf{y}^k$ by calculating $\delta(k)$ by (3.3), instead of calculating directly the values $f(\mathbf{y}^k)$, which reduces sharply the computational cost. By Definition 2.1, there exist two points satisfying (3.1) for fixed k; that is, when $\mathbf{y}^k \in N(\mathbf{y}^0)$ and (3.1) is satisfied, then either $y_k^k = \tau_1(y_k^0)$ or $y_k^k = \tau_2(y_k^0)$. For our convenience, we denote $\delta(k)$ by $\delta_1(k)$ when $y_k^k = \tau_1(y_k^0)$ and by $\delta_2(k)$ when $y_k^k = \tau_2(y_k^0)$. In what follows, we describe the local search algorithm for the Max 3-cut problem denoted by LSM3C; by this algorithm, we can get a local maximizer of function $f(\mathbf{y})$ over $\mathbb{S}$.

For **LSM3C**, one has the following.

(1) Input any initial feasible solution $\mathbf{y}^0$ of problem (M3C).

(2) For k from 1 to n, set $\mathbf{z}^{k1} = \tau_1^k(\mathbf{y}^0)$, calculate $\delta_1(k)$, and set again $\mathbf{z}^{k2} = \tau_2^k(\mathbf{y}^0)$, calculate $\delta_2(k)$.

(3) Find $\delta_{i^*}(k^*)$ by the following way:

$$\delta_{i^*}(k^*) = \min\{\delta_1(1), \delta_2(1), \ldots, \delta_1(k), \delta_2(k), \ldots, \delta_1(n), \delta_2(n)\}. \tag{3.4}$$

(4) If $\delta_{i^*}(k^*) \geq 0$, then set $\hat{\mathbf{y}} = \mathbf{y}^0$, return $\hat{\mathbf{y}}$, and stop. Otherwise, go to next.

(5) Set $\mathbf{y}^0 = \tau_{i^*}^{k^*}(\mathbf{y}^0)$; go to Step 2.

3.2. Variable Neighborhood Stochastic Search

Let $\hat{\mathbf{y}}$ be a local maximizer obtained by LSM3C and $K_{\max}$ $(1 < K_{\max} \leq n)$ a fixed positive integer number. we now describe the variable neighborhood search (VNS) with stochastic steps, by which we can find an approximate global maximizer of problem (M3C). The proposed VNS algorithm actually has three phases: First, for any given positive integer number $K < K_{\max}$, a K-neighborhood point, $\mathbf{y}$ say, is randomly selected; that is, $\mathbf{y} \in N_K(\hat{\mathbf{y}})$. Next, a solution, $\hat{\hat{\mathbf{y}}}$ say, is obtained by applying algorithm LSM3C to $\mathbf{y}$. Finally, the current solution jumps from $\hat{\mathbf{y}}$ to $\hat{\hat{\mathbf{y}}}$ if it improves the former one. Otherwise, the order K of the neighborhood is increased by one when $K < K_{\max}$ and the above steps are repeated until some stopping condition is met. The VNS that is also called k-max [11] can be illustrated as follows.

For **VNS-k**, one has the following.

(1) Arbitrary choose a point $\mathbf{y}^0 \in \mathbb{S}$, implement LSM3C starting from $\mathbf{y}^0 \in \mathbb{S}$ and denote the obtained local maximizer by $\hat{\mathbf{y}}$. Set $K = 1$.

(2) Randomly take a point $\mathbf{y} \in \partial N_{I(K)}(\hat{\mathbf{y}})$ and implement again LSM3C from $\mathbf{y}$, and denote the obtained new local maximizer by $\hat{\hat{\mathbf{y}}}$.

(3) If $f(\hat{\hat{\mathbf{y}}}) > f(\hat{\mathbf{y}})$, then set $\hat{\mathbf{y}} = \hat{\hat{\mathbf{y}}}$ and $K = 1$; go to Step 2.

(4) If $K < K_{\max}(\leq n)$, set $K = K + 1$; go to Step 2. Otherwise, return $\hat{\mathbf{y}}$ as an approximate global solution of problem M3C and stop.

The subscript $I(K)$ in Step 2 is a function of K and is also a positive integer number not greater than n. $I(K)$ reflects the main skill of converting the current neighborhood of local maximizer $\hat{\mathbf{y}}$ into another neighborhood of $\hat{\mathbf{y}}$. For a given $K_{\max}$, let $m = \lfloor n/K_{\max} \rfloor$ and $K_0 = n - mK_{\max}$, where $\lfloor a \rfloor$ means the integral part of a. We divide the n neighborhoods of $\hat{\mathbf{y}}$, $N(\hat{\mathbf{y}}), N_2(\hat{\mathbf{y}}), \ldots, N_K(\hat{\mathbf{y}}), \ldots, N_n(\hat{\mathbf{y}})$ into $K_{\max}$ neighborhood blocks $N_{I(1)}(\hat{\mathbf{y}}), \ldots, N_{I(K_{\max})}(\hat{\mathbf{y}})$, such that, for $K = 1, 2, \ldots, K_{\max} - K_0$,

$$N_{(K-1)m+1}(\hat{\mathbf{y}}) \subseteq N_{I(K)}(\hat{\mathbf{y}}) \subseteq N_{Km+1}(\hat{\mathbf{y}}), \tag{3.5}$$

and, for $K = K_{\max} - K_0 + 1, \ldots, K_{\max} - K_0 + j, \ldots, K_{\max}$,

$$N_{(K-1)(m+1)+1}(\hat{\mathbf{y}}) \subseteq N_{I(K)}(\hat{\mathbf{y}}) \subseteq N_{K(m+1)}\ (\hat{\mathbf{y}}). \tag{3.6}$$

In order to obtain the $K_{\max}$ neighborhood blocks of $\hat{\mathbf{y}}$, $N_{I(K)}(\hat{\mathbf{y}})$, $K = 1, \ldots, K_{\max}$, we divide the set $\{1, 2, \ldots, n\}$ into $K_{\max}$ disjoint subsets, where each subset of the first $K_{\max} - K_0$ subsets

has m integers and each subset of the last K_0 subsets has $m+1$ integers. For any integer $K(\leq K_{\max})$, let

$$I(K) = (K-1)\cdot m + [c\cdot m] + 1, \quad K = 1,2,\ldots,K_{\max} - K_0, \tag{3.7}$$

or

$$\begin{aligned} I(K) &= (K_{\max} - K_0)m + [(m+1)\cdot c] \ + 1 + (m+1)(K - (K_{\max} - K_0) - 1) \\ &= [(m+1)\cdot c] + 1 + (m+1)(K-1) - (K_{\max} - K_0), \\ K &= K_{\max} - K_0 + 1, \ldots, K_{\max} - K_0 + j, \ldots, K_{\max}. \end{aligned} \tag{3.8}$$

Then we can randomly choose a point $\mathbf{y}$ in $\partial N_{I(K)}(\hat{\mathbf{y}})$, where $c \in (0,1)$ is a random number from uniformly distribution $\mathcal{U}(0,1)$, such that $N_{I(K)}(\hat{\mathbf{y}})$ satisfies (3.5) or (3.6).

VNS-k stops when the maximum K neighborhood is reached. Additionally, we also consider another termination criterion of VNS based on the maximum CPU-time and denoted by VNS-t. VNS-t can obtain a better solution than VNS-k since VNS-t actually runs several times VNS-k in the maximum allowing time $t_{\max}$, but it generally has to spend more computational time. The VNS-t can be stated as follows.

For **VNS-t**, one has the following.

(1) Set $t_{\mathrm{CPU}} = 0$, running VNS-k for an arbitrary initial point $\mathbf{y}^0 \in \mathbb{S}$, and let a local optimal solution $\hat{\mathbf{y}}$ be obtained.

(2) If $K = K_{\max}(\leq n)$, go to Step 3.

(3) If $t_{\mathrm{CPU}} < t_{\max}$, then set $K = 1$; go to Step 2 in VNS-k. Otherwise, return $\hat{\mathbf{y}}$ as an approximate global solution of problem M3C and stop.

We mention that it differs from the classical variable neighborhood search metaheuristic that is originally proposed by Mladenović and Hansen [10]. In order to obtain a global optimal solution or a high-quality approximate solution of problem M3C, we use two stochastic steps in VNS. First, for a fixed K, a K-neighbor of $\hat{\mathbf{y}}$ is chosen randomly. Second, by the definition of $I(K)$, when we change the neighborhood of $\hat{\mathbf{y}}$ from $N_{I(K-1)}$ to $N_{I(K)}$, $N_{I(K)}$ may take any a neighborhood among $N_{(K-1)m+j}$, $j = 1,2,\ldots,m$ of $\hat{\mathbf{y}}$, which is decided by random number c. In VNS, positive integer $K_{\max}$ decides the maximum search neighborhood block of $\hat{\mathbf{y}}$, which also decides directly the CPU-time of VNS. Based on the second stochastic step, we may choose a relative small $K_{\max}$ comparing with n. This can decrease our computational time.

4. A Greedy Algorithm for Max 3-Section

When the number of nodes n is a multiple of three and the condition $|S_0| = |S_1| = |S_2| = n/3$ is required, the Max 3-cut problem becomes the Max 3-section problem. Notice that $1+\omega+\omega^2 =$

0, then the Max 3-section problem can be formulated as the following programming problem M3S:

$$\begin{aligned} \text{M3S}: \max \quad & \frac{2}{3}\sum_{i<j} w_{ij}\left(1-\operatorname{Re}\left(y_i\cdot y_j\right)\right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i = 0, \\ & \mathbf{y}\in\mathbb{S}, \end{aligned} \tag{4.1}$$

and its CSDP relaxation is

$$\begin{aligned} \text{CSDP1}: \max \quad & \frac{2}{3}\sum_{i<j} w_{ij}\left(1-\operatorname{Re}\left(\mathbf{y}_i\cdot \mathbf{y}_j\right)\right) \\ \text{s.t.} \quad & \mathbf{e}\mathbf{e}^T\cdot Y = 0, \\ & \|\mathbf{y}_i\| = 1,\ i = 1,2,\ldots,n, \\ & A_{ij}^k\cdot Y \geq -1, \quad i,j = 1,2,\ldots,n,\ k = 0,1,2 \\ & Y \succcurlyeq 0, \end{aligned} \tag{4.2}$$

where $\mathbf{e}$ is the column vector of all ones. Andersson [12] extended Frieze and Jerrum's random rounding method to M3S and obtained a $(2/3) + O(1/n^3)$-approximate algorithm, which is the current best approximate ratio for M3S; also see the recent research of Gaur et al. [13]. The author of the current paper considers a special the Max 3-Section problem and obtains a 0.6733-approximate algorithm; see Ling (2009) [14].

Clearly, the feasible region of problem M3S is a subset of $\mathbb{S}$, and the optimal value of problem M3S is not greater than that of problem M3C. Assume that we have get a global optimal solution or a high-quality approximate solution $\widehat{\mathbf{y}}$ of problem M3C. It is clear that $\widehat{\mathbf{y}}$ may not satisfy the condition $\sum_{i=1}^{n}\widehat{y}_i = 0$. But we may adjust $\widehat{\mathbf{y}}$ to get a new feasible solution $\mathbf{y}^s$ using a greedy algorithm, such that $\mathbf{y}^s$ satisfies $\sum_{i=1}^{n} y_i^s = 0$. This is the motivation that we propose the greedy algorithm for the Max 3-section problem.

For the sake of our analysis, without loss of generality, we assume that the local maximizer $\widehat{\mathbf{y}}$ satisfies $|S_0| = \max\{|S_0|,|S_1|,|S_2|\}$. This means that $S_0 = \{i : \widehat{y}_i = 1\}$ is the subset of V with maximum cardinal number. If $|S_k| = \max\{|S_0|,|S_1|,|S_2|\}(k\neq 0, k = 1,2)$, then we may set $y_i^N = \overline{w}^k\widehat{y}_i$, $i = 1,\ldots,n$. The resulted new solution $\mathbf{y}^N = (y_1^N,\ldots,y_i^N)$ will not change the objective value since $f(\widehat{\mathbf{y}}) = f(\overline{w}^k\widehat{\mathbf{y}})$ $(k\neq 0, k = 1,2)$; moreover, the new partition $\{S_0^N, S_1^N, S_2^N\}$ based on $\mathbf{y}^N$ satisfies $|S_0^N| = \max\{|S_0^N|,|S_1^N|,|S_2^N|\}$. By our assumption, the partition $S = \{S_0, S_1, S_2\}$ still exist four possible cases.

Case 1. $|S_0| \geq |S_1| \geq n/3 \geq |S_2|$.

Case 2. $|S_0| \geq n/3 \geq |S_1| \geq |S_2|$.

Case 3. $|S_0| \geq |S_2| \geq n/3 \geq |S_1|$.

Case 4. $|S_0| \geq n/3 \geq |S_2| \geq |S_1|$.

The sizes adjusting greedy algorithm of Cases 3 and 4 are similar to Cases 1 and 2. Hence, we mainly consider Cases 1 and 2 for adjusting the partition of V from $S = \{S_0, S_1, S_2\}$ to $\widetilde{S} = \{\widetilde{S}_0, \widetilde{S}_1, \widetilde{S}_2\}$ such that $|\widetilde{S}_k| = n/3$, $k = 0, 1, 2$. Denote

$$\begin{aligned}
\delta_0(i) &= \sum_{j\in S_1\cup S_2} w_{ij}, \quad i \in S_0,\\
\delta_{01}(i) &= \sum_{j\in S_1} w_{ij}, \quad i \in S_0, \qquad \delta_{10}(i) = \sum_{j\in S_0} w_{ij}, \quad i \in S_1,\\
\delta_{02}(i) &= \sum_{j\in S_2} w_{ij}, \quad i \in S_0, \qquad \delta_{20}(i) = \sum_{j\in S_0} w_{ij}, \quad i \in S_2,\\
\delta_{12}(i) &= \sum_{j\in S_2} w_{ij}, \quad i \in S_1, \qquad \delta_{21}(i) = \sum_{j\in S_1} w_{ij}, \quad i \in S_2.
\end{aligned} \tag{4.3}$$

Then, it follows from simple computation that

$$\begin{aligned}
&\delta_0(i) = \delta_{01}(i) + \delta_{02}(i), \quad \text{for each } i \in S_0,\\
&\sum_{i\in S_k} \delta_{kl}(i) = \sum_{i\in S_l} \delta_{lk}(i), \quad k, l = 0,\ 1,\ 2,\ k \neq l,\\
&f(\hat{\mathbf{y}}) = \sum_{i\in S_0} \delta_0(i) + \sum_{i\in S_1} \delta_{12}(i)\\
&\qquad = \sum_{i\in S_0} \delta_{01}(i) + \sum_{i\in S_0} \delta_{02}(i) + \sum_{i\in S_1} \delta_{12}(i)\\
&\qquad = d_{01} + d_{02} + d_{12},
\end{aligned} \tag{4.4}$$

where $d_{01} = \sum_{i\in S_0} \delta_{01}(i)$, $d_{02} = \sum_{i\in S_0} \delta_{02}(i)$, $d_{12} = \sum_{i\in S_1} \delta_{12}(i)$.

In what follows, we describe the size adjusting greedy algorithms (SAGAs) for Cases 1 and 2, and denote the greedy algorithms for the two cases by SAGA1 and SAGA2, respectively.

For **SAGA1**, one has the following.

(1) Calculate

$$m_{02} = \frac{\sum_{i\in S_0} \delta_{02}(i)}{|S_0|}, \qquad m_{12} = \frac{\sum_{i\in S_1} \delta_{12}(i)}{|S_1|}. \tag{4.5}$$

(2) If $m_{02} \geq m_{12}$, let $S_1 = \{j_1, j_2, \ldots, j_{|S_1|}\}$, where $\delta_{12}(j_l) \geq \delta_{12}(j_{l+1})$, $l = 1, 2, \ldots, |S_1|$. Set $\widetilde{S}_1 = \{j_1, j_2, \ldots, j_{n/3}\}$, $\widetilde{S}_2 = S_2 \bigcup (S_1 \setminus \widetilde{S}_1)$ and renew to calculate

$$\delta'_{02}(i) = \sum_{j\in \widehat{S}_2} w_{ij}, \tag{4.6}$$

for each $i \in S_0$. Let $S_0 = \{i_1, i_2, \ldots, i_{|S_0|}\}$, where $\delta'_{02}(i_k) \geq \delta'_{02}(i_{k+1})$. Set $\widetilde{S}_0 = \{i_1, i_2, \ldots, i_{n/3}\}$ and $\widetilde{S}_2 = \widetilde{S}_2 \bigcup (S_0 \setminus \widetilde{S}_0)$.

(3) If $m_{02} < m_{12}$, let $S_0 = \{i_1, i_2, \ldots, i_{|S_0|}\}$, where $\delta_{02}(i_k) \geq \delta_{02}(i_{k+1})$, $k = 1, 2, \ldots, |S_0|$, set $\widetilde{S}_0 = \{i_1, i_2, \ldots, i_{n/3}\}$, $\widehat{S}_2 = S_2 \bigcup (S_0 \setminus \widetilde{S}_0)$, and then renew to calculate

$$\delta'_{12}(i) = \sum_{j \in \widehat{S}_2} w_{ij}, \tag{4.7}$$

for each $i \in S_1$. Set $\widetilde{S}_1 = \{j_1, j_2, \ldots, j_{n/3}\}$ and $\widetilde{S}_2 = \widehat{S}_2 \bigcup (S_1 \setminus \widetilde{S}_1)$, where $\delta'_{12}(j_k) \geq \delta'_{12}(j_{k+1})$ here.

(4) Return the current partition $\widetilde{S} = \{\widetilde{S}_0, \widetilde{S}_1, \widetilde{S}_2\}$; stop.

For **SAGA2**, one has the following.

(1) Calculate $d_{01} = \sum_{i \in S_0} \delta_{01}(i)$, $d_{02} = \sum_{i \in S_0} \delta_{02}(i)$, and

$$m_{01} = \frac{d_{01}}{|S_0|}, \qquad m_{02} = \frac{d_{02}}{|S_0|}. \tag{4.8}$$

(2) If $m_{01} \leq m_{02}$, let

$$S_0 = \{i_1, i_2, \ldots, i_{|S_0|}\}, \tag{4.9}$$

where $\delta_{01}(i_k) \geq \delta_{01}(i_{k+1})$, $k = 1, 2, \ldots, |S_0|$. Set

$$\widehat{S}_0 = \{i_1, i_2, \ldots, i_{|S_0|-q_1}\}, \qquad \widetilde{S}_1 = S_1 \bigcup \left(S_0 \setminus \widehat{S}_0\right), \tag{4.10}$$

where $q_1 = (n/3) - |S_1|$. Renew to calculate

$$\delta'_{02}(i) = \sum_{j \in S_2} w_{ij}, \quad i \in \widehat{S}_0. \tag{4.11}$$

and let

$$\widehat{S}_0 = \left\{i'_1, i'_2, \ldots, i'_{|\widehat{S}_0|}\right\}, \tag{4.12}$$

where $\delta'_{02}(i'_k) \geq \delta'_{02}(i'_{k+1})$, $k = 1, 2, \ldots, |\widehat{S}_0|$. Set

$$\widetilde{S}_0 = \left\{i'_1, i'_2, \ldots, i'_{n/3}\right\}, \qquad \widetilde{S}_2 = S_2 \bigcup \left(\widehat{S}_0 \setminus \widetilde{S}_0\right). \tag{4.13}$$

(3) If $m_{01} > m_{02}$, let

$$S_0 = \{i_1, i_2, \ldots, i_{|S_0|}\}, \tag{4.14}$$

where $\delta_{02}(i_k) \geq \delta_{02}(i_{k+1})$, $k = 1, 2, \ldots, |S_0|$. Set

$$\widehat{S}_0 = \{i_1, i_2, \ldots, i_{|S_0|-q_2}\}, \qquad \widetilde{S}_2 = S_2 \bigcup \left(S_0 \setminus \widehat{S}_0\right), \tag{4.15}$$

where $q_2 = (n/3) - |S_2|$. Renew to calculate

$$\delta'_{01}(i) = \sum_{j \in S_1} w_{ij}, \quad i \in \widehat{S}_0. \tag{4.16}$$

and let

$$\widehat{S}_0 = \left\{i'_1, i'_2, \ldots, i'_{|\widehat{S}_0|}\right\}, \tag{4.17}$$

where $\delta'_{01}(i'_k) \geq \delta'_{01}(i'_{k+1})$, $k = 1, 2, \ldots, |\widehat{S}_0|$. Set

$$\widetilde{S}_0 = \left\{i'_1, i'_2, \ldots, i'_{n/3}\right\}, \qquad \widetilde{S}_1 = S_1 \bigcup \left(\widehat{S}_0 \setminus \widetilde{S}_0\right). \tag{4.18}$$

(4) Return the current partition $\widetilde{S} = \{\widetilde{S}_0, \widetilde{S}_1, \widetilde{S}_2\}$; stop.

5. Numerical Results

This section describes the obtained experimental results for some instances of Max 3-cut and Max 3-Section problems using the proposed VNS metaheuristic. We also show a quantitative comparison with 0.836-approximate algorithm. The computational experiments are performed in an Intel Pentium 4 processor at 2.0 GHz, with 512 MB of RAM, and all algorithms are coded in Matlab. Because RSDP relaxation of M3C includes many slack variables, many constraints, and matrices variables without a block diagonal structures, in our numerical comparisons, we choose SDPT3-4.0 [9], one of the best and well-known solvers of semidefinite programming, to solve RSDP relaxation of M3C.

All our test problems are generated randomly by the following way. Let $p \in (0, 1)$ be a constant and $r \in (0, 1)$ a random number. If $r \leq p$, then there is an edge between nodes i and j with weight w_{ij}, that is, a random integer between 1 and 10. Otherwise, $w_{ij} = 0$; that is, there is no edge between nodes i and j. Because of the limits of memory of SDPT3, when $n > 200$, RSDP becomes a huge semidefinite programming problem with not less than 59700 slack variables and 99900 constraints and is out of memory of SDPT3. Hence, in the numerical experiments, we consider 30 instances with $p = 0.1, 0.3, 0.6$, and n varying from 20 to 200.

Firstly, we check the influence of $K_{\max}$ on the quality of solution obtained by VNS-k. For a given graph, we take $K_{\max} = 3, 5, 10, 15, 30$; Table 1 presents the results, where `Wnp` in the first column of this table and the following tables means that a graph is randomly generated with nodes n and density p; for instance, W30.6 presents a graph generated randomly with $n = 30$ and $p = 0.6$. We find from Table 1 that the influence of $K_{\max}$ to objective value denoted by `Obj` in Table 1 is slight when $K_{\max} > 5$, but the CPU time increases sharply as $K_{\max}$ increases. This result is actually not surprising. Indeed, because $I(K) > K$, we choose

Table 1: The objective value obtained by VNS for M3C with different $K_{\max}$.

Wnp	$K_{\max} = 3$		$K_{\max} = 5$		$K_{\max} = 10$		$K_{\max} = 15$		$K_{\max} = 30$	
	Obj	t	Obj	t	Obj	t	Obj	t	Obj	t
W60.1	960	1.38	965	2.14	967	4.63	967	6.29	969	12.50
W100.3	6001	7.55	6010	12.66	6013	23.37	6013	29.28	6015	69.52
W120.1	3323	13.76	3335	34.07	3337	38.78	3339	50.48	3343	101.85

randomly a point $\mathbf{y}$ in $\partial N_{I(K)}(\hat{\mathbf{y}})$, instead of $\partial N_K(\hat{\mathbf{y}})$. This avoids to choose too large $K_{\max}$ which leads to more CPU-time cost. Hence, in sequel numerical comparisons, we fix $K_{\max} = 5$ for all test problems.

Secondly, we compare VNS (VNS-k, VNS-t) metaheuristic with 0.836-approximate algorithm for all test problems. To avoid the effect of initial points, for each test problem, after RSDP is solved, we run the round procedure of 0.836-approximate algorithm and VNS metaheuristic ten times, respectively.

Table 2 gives the result of numerical comparisons. In the numerical presentations of Table 2, $\mathtt{Obj}_{\mathrm{rsdp}}$ is the optimal value of problem RSDP; that is, it is an upper bound of M3C. $\mathtt{Obj}_{\mathrm{GM}}$ is the largest value obtained by 0.836-approximate algorithm in the ten tests. $\mathtt{Obj}_{\mathrm{vns}}$ stands for the largest value obtained by VNS for M3C in the ten tests, respectively. m and $\mathtt{s.v.}$ are the number of constraints and slack variables (s.v.), respectively. t_{GM} and $t_{\mathrm{vns}-k}$ are the average time (second) associated with the two algorithms in the ten tests. For the maximum CPU time of VNS-t, we take $t_{\max} = 2t_{\mathrm{vns}-k}$, but the real CPU time of VNS-t will be greater than $t_{\max}$. Additionally, for measuring the performance of solutions, we take

$$\rho = \frac{\mathtt{Obj}_{\mathrm{vns}} - \mathtt{Obj}_{\mathrm{rsdp}}}{\mathtt{Obj}_{\mathrm{rsdp}}} = \frac{\mathtt{Obj}_{\mathrm{vns}}}{\mathtt{Obj}_{\mathrm{rsdp}}} - 1 \tag{5.1}$$

for M3C and

$$\rho = \frac{\mathtt{Obj}_{\mathrm{vns+saga}} - \mathtt{Obj}_{\mathrm{rsdp}}}{\mathtt{Obj}_{\mathrm{rsdp}}} = \frac{\mathtt{Obj}_{\mathrm{vns+saga}}}{\mathtt{Obj}_{\mathrm{rsdp}}} - 1 \tag{5.2}$$

for M3S. Clearly, ρ can reflect how close to the solution obtained by VNS from the optimal solution of RSDP. One can see from Table 2 that (1) the VNS metaheuristic not only can obtain a better solution than 0.836-approximate algorithm for all test problems, but also that the elapsed CPU-time of VNS metaheuristic is much less than that of 0.836-approximate algorithm for all test problems, (2) the performance of solution can be improved by VNS-t for most of test problems when the termination criterion of VNS is based on the maximum CPU-time, but VNS-t spends more computational time than VNS-k. The improved performance can be reflected by $\nabla\rho = \rho_t - \rho_k$ in the final column of Table 2. Average speaking, VNS-t improves 0.91 percentage point.

Finally, we consider the solution of M3S by combining VNS-k and greedy sizes-adjusted algorithm SAGA stated in Section 4. Let $\hat{\mathbf{y}}$ be an approximate solution of M3C obtained by VNS; we can obtain an approximate solution of M3S from SAGA. The numerical results are reported by Table 3 in which $\mathtt{Obj}_{\mathtt{vns+saga}}$ stands for the largest value obtained by VNS-k plus SAGA for M3S. Although our sizes-adjusted algorithm may decrease the objective value obtained by VNS, the changes of objective values are very slight from

Table 2: The numerical comparisons of 0.836-approximate algorithm with VNS metaheuristic.

Wnp	m	$\mathtt{Obj}_{\mathrm{rsdp}}$	0.836-algorithm		VNS					
	s.v.		$\mathtt{Obj}_{\mathrm{GM}}$	t_{GM}	$\mathtt{Obj}_{\mathrm{vns}\text{-}k}$	$t_{\mathrm{vns}\text{-}k}$	$\rho_k\%$	$\mathtt{Obj}_{\mathrm{vns}-t}$	$\rho_t\%$	$\nabla\rho\%$
W20.1	990	121	117	22.93	119	0.28	−1.65	119	−1.65	0
W20.3	570	232	225	25.58	228	0.29	−1.72	228	−1.72	0
W20.6		492	464	14.27	479	0.40	−2.65	479	−2.24	0.41
W30.1	2235	147	139	161.56	144	0.81	−2.04	144	−2.04	0
W30.3	1305	566	521	140.12	550	0.97	−2.83	550	−2.30	0.53
W30.6		1181	1101	94.93	1151	0.92	−2.54	1151	−2.03	0.51
W45.1	5040	675	594	202.48	605	0.92	−10.38	612	−9.33	1.05
W45.3	2970	1300	1147	211.72	1192	1.02	−8.31	1300	−7.15	1.16
W45.6		2441	2272	217.57	2313	1.02	−5.25	2350	−3.73	1.52
W60.1	8970	1069	918	256.35	965	2.14	−9.73	981	−8.23	1.50
W60.3	5310	2494	2228	256.57	2301	3.13	−7.74	2332	−6.50	1.24
W60.6		4403	4054	266.68	4213	2.48	−4.32	4307	−2.18	2.14
W80.1	15960	1567	1357	537.58	1415	5.82	−9.71	1438	−8.23	1.48
W80.3	9480	4094	3666	546.23	3801	7.35	−7.16	3912	−4.45	2.71
W80.6		7746	7328	495.34	7423	6.31	−4.17	7481	−3.42	0.75
W100.1	24950	2526	2149	893.52	2262	12.73	−10.46	2302	−8.87	1.60
W100.3	14850	6418	5814	882.73	6010	12.66	−6.36	6113	−4.75	1.61
W100.6		11956	11212	865.28	11391	11.99	−4.73	11422	−4.47	0.26
W120.1	33990	3746	3222	1476.54	3335	34.07	−10.98	3392	−9.45	1.53
W120.3	21420	9173	8266	1498.18	8575	35.41	−6.52	8623	−6.00	0.52
W120.6		16748	15596	1567.43	16056	43.80	−4.14	16114	−3.79	0.35
W150.1	56175	5821	5020	2618.11	5208	69.20	−10.54	5257	−9.69	0.85
W150.3	33525	14432	13209	3021.87	13543	70.60	−6.15	13607	−5.72	0.43
W150.6		26310	25076	3172.22	25405	74.46	−3.44	25517	−3.01	0.43
W180.05	80910	4303	3578	9043.25	3726	133.10	−13.41	3812	−11.41	1.20
W180.1	48330	7236	6328	10225.54	6481	130.07	−10.44	6547	−9.52	0.92
W180.3		20147	18436	9887.36	19031	132.14	−5.54	19213	−4.64	0.90
W180.6		37292	35386	10004.11	35949	102.77	−3.61	36124	−3.13	0.48
W200.05	99900	5174	4306	25872.33	4484	71.90	−13.40	4509	−12.85	0.55
W200.1	59700	9271	8092	29003.28	8799	149.50	−5.10	8853	−4.51	0.59
W200.5		38831	36481	28774.17	37477	200.16	−3.49	37552	−3.29	0.20

Table 3. Particular, objective values of some problems do not decrease, instead increase, such as W150.3. We do not compare the obtained results with Andersson's 2/3-approximate algorithm. Because we find that all approximate solutions of M3S obtained by VNS plus SAGA still are better than that of 0.836-approximate algorithm with the exception of only W30.1 and W30.3.

6. Conclusions

A variable neighborhood stochastic metaheuristic was proposed to solve the Max 3-cut and Max 3-section problems in this paper. Our algorithms can solve Max 3-cut and Max 3-section problems with different sizes and densities. Although 0.836-approximate algorithm

Table 3: The numerical results of combining VNS-k metaheuristic with SAGA for M3S.

Wnp	$\mathtt{Obj}_{\mathrm{rsdp}}$	$\mathtt{Obj}_{\mathrm{GM}}$	$\mathtt{Obj}_{\mathrm{vns+saga}}$	ρ_k (%)
W30.1	147	139	138	−6.12
W30.3	566	521	518	−8.49
W30.6	1181	1101	1151	−2.55
W45.1	675	594	605	−10.38
W45.3	1300	1147	1191	−8.39
W45.6	2441	2272	2313	−5.25
W60.1	1069	918	952	−10.95
W60.3	2494	2228	2301	−7.74
W60.6	4403	4054	4213	−4.32
W120.1	3746	3222	3290	−12.18
W120.3	9173	8266	8449	−7.90
W120.6	16748	15596	16056	−4.14
W150.1	5821	5020	5203	−10.62
W150.3	14432	13209	13551	−6.11
W150.6	26310	25076	25096	−4.62
W180.05	4303	3578	3726	−13.41
W180.1	7236	6328	6396	−11.61
W180.3	20147	18436	18823	−6.58
W180.6	37292	35386	35947	−3.61

has the very good theoretic results, in numerical aspects, our comparisons indicate that the proposed VNS metaheuristic is superior to the well-known 0.836-approximate algorithm and can efficiently obtain very high-quality solutions of the Max 3-cut and Max 3-section problems.

We mention that the proposed algorithm in fact can deal with higher dimensional G-set graphs problems created by Pro. Rinaldi using a graph generator, rudy. But, we cannot give numerical comparisons with 0.836-approximate algorithm since RSDP relaxations of these problems are out of memory of the current all SDP software. In additionally, if we increase $K_{\max}$ or $t_{\max}$ in numerical implementing, then the quality of solution of M3C will further be improved by VNS.

Funding

This work is supported by the National Natural Science Foundations of China (no. 71001045, 10971162), the China Postdoctoral Science Foundation (no. 20100480491), the Natural Science Foundation of Jiangxi Province of China (no. 20114BAB211008), and the Jiangxi University of Finance and Economics Support Program Funds for Outstanding Youths.

Acknowledgment

The authors would like to thank the editor and an anonymous referee for their numerous suggestions for improving the paper.

References

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds., pp. 85–103, Plenum Press, New York, NY, USA, 1972.

[2] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237–267, 1976.

[3] F. Barahona, M. Gröetschel, G. Reinelt, and M. Juenger, "An application of combinotiorial optimization to statistical physics and circuit layout design," *Operations Research*, vol. 36, no. 3, pp. 493–513, 1988.

[4] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the Association for Computing Machinery*, vol. 42, no. 6, pp. 1115–1145, 1995.

[5] A. Frieze and M. Jerrum, "Improved approximation algorithms for MAX k-CUT and MAX BISECTION," in *Integer Programming and Combinatorial Optimization*, E. Balas and J. Clausen, Eds., vol. 920, pp. 1–13, 1995.

[6] M. X. Goemans and D. P. Williamson, "Approximation algorithms for MAX-3-CUT and other problems via complex semidefinite programming," *Journal of Computer and System Sciences*, vol. 68, no. 2, pp. 442–470, 2004.

[7] S. Zhang and Y. Huang, "Complex quadratic optimization and semidefinite programming," *SIAM Journal on Optimization*, vol. 16, no. 3, pp. 871–890, 2006.

[8] J. F. Sturm, "Using SeDuMi 1.02, 'a MATLAB toolbox for optimization over symmetric cones'," *Optimization Methods and Software*, vol. 11, no. 1–4, pp. 625–653, 1999.

[9] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, "SDPT3 version 4.0 (beta)—a MATLAB software for semidefinite-quadratic-linear programming," 2004, http://www.math.nus.edu.sg/~mattohkc/sdpt3.html.

[10] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[11] P. Hansen, N. Mladenović, and J. A. Moreno Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, pp. 367–407, 2010.

[12] G. Andersson, "An approximation algorithm for Max p-Section," *Lecture Notes in Computer Science*, vol. 1563, pp. 237–247, 1999.

[13] D. R. Gaur, R. Krishnamurti, and R. Kohli, "The capacitated max k-cut problem," *Mathematical Programming*, vol. 115, no. 1, pp. 65–72, 2008.

[14] A.-F. Ling, "Approximation algorithms for Max 3-section using complex semidefinite programming relaxation," in *Combinatorial Optimization and Applications*, vol. 5573 of *Lecture Notes in Computer Science*, pp. 219–230, 2009.