基于遗传算法及贪心策略的多波束测深模型

摘要

多波束测深在地形平坦情况下可测量出以测量船测线为轴线且具有一定宽度的全覆盖水深条带，在海洋地形测绘领域有极高的应用价值。本文基于几何学以及运筹学相关知识，建立了多波束测深的覆盖宽度及相邻条带重叠率的数学模型，给出了海底是斜坡的情况下可完全覆盖整个待测海域的最短测线；同时根据矩形分块的方法给出了一般海域下的测线方案。

针对问题 1，对于测线与坡面平行的特殊情况，我们在斜坡侧视图中进行分析，借助**正弦定理及相似三角形**等平面几何知识，用相邻条带间距表示覆盖宽度，代入重叠率的定义可得相邻条带之间重叠率关于相邻条带间距的函数表达式，从而建立了**测线平行于海底坡面情况下的多波束测深模型**。通过 python 求解，计算测线距中心点处的不同距离时海水深度、覆盖宽度以及与前一条测线的重叠率，将结果保存为 result1.xlsx 文件。

针对问题 2，对于测线与坡面存在夹角的情况，借助**空间立体几何**的知识，给出了垂直于测线的平面与海底坡面的交线的方向向量表达式，通过空间向量的夹角控制，确定了该交线与水平面的夹角。通过考虑包含该覆盖区域的所在平面，将本问化归回问题 1 中的模型，求解得测线上任意一点的覆盖宽度关于测量船距海域中心点处的距离及测线方向夹角的函数表达式，建立了**测线不平行于海底坡面情况下的多波束测深模型**。通过 python 求解，将所得结果保存在 result2.xlsx 中。

针对问题 3，结合问题 2 中的模型，定义测线上任意一点与相邻测线的间距为点到直接的距离，给出了相邻测线不平行时一般情况的重叠率表达式。以测线长度为优化目标，以覆盖整个海域为绝对约束，考虑重叠率范围的约束条件，建立了**非线性规划模型**。由于常规的求解器求解效率较低，考虑设计**遗传算法**求其近似解。将各条测线与坡面法向在水平面上的投影方向的夹角编入染色体编码中，以测量长度的倒数为适应度函数，采取轮盘赌的方式进行选择，按照一定的概率进行单点交叉和单点变异，直至达到最大代数或所有个体的适应度不再上升时，求得**可完全覆盖整个待测海域的测线的最短测量长度为 $114824.814m$，总共包含 31 条测线**，同时计算测线夹角为 90° 时的最短测线长度，可发现上述结果更优。

针对问题 4，针对地形不平坦的情况，需考虑测线总长度、测线长度超过总长度部分、未测海域占比三个优化目标，结合问题 2 及问题 3 的模型，对海域进行矩形分块，通过**贪心策略**进行相邻矩形分块的重叠率的选择，对于重叠率加以范围的限制转化为约束，通过仿真以**标记散点**的方式计算未测海域占比，赋予未测海域占比最高的优先因子，离散化处理该范围中的重叠率遍历得到结果，同时考虑余下两个优化目标，其中较优情况为：**测线总长度为 $442371.69m$，漏测面积占比为 2.13%，重叠率超过 20% 的部分长度为 $4527m$**。

关键词：多波束测深　非线性规划模型　遗传算法　贪心策略　多目标优化

一、问题重述

1.1 问题背景

单波束测深是海洋测绘工程中的常用技术，利用声波在均匀介质中沿直线传播并在不同界面上发生反射的原理，可通过声波在海水中的传播时间和传播速度计算出海水深度。受限于单点连续的测量方法，单波束测深沿航迹间的数据十分密集，但在测线间没有数据。多波束测深系统在单波束测深的基础上进行了改进，它可以一次性发射多个波束，形成一个波束阵列，以测量船测线为轴线并具有一定宽度的全覆盖水深条带，如图 1 所示。

多波束测深条带的覆盖宽度 W 与换能器开角 θ 和水深 D 相关。在测线相互平行且海底地形平坦的前提下，相邻条带之间的重叠率可定义为 $\eta = 1 - \frac{d}{W}$，其中 d 为相邻两条测线的间距，W 为条带的覆盖宽度，如图 2。重叠率为负数，则表示漏测。相邻条带之间需有 10% ~20% 的重叠率以保证测量的便利性和数据的完整性。由于真实海底地形崎岖，则需考虑重叠率与漏测现象间的制约关系：采取海水平均水深设计测线间隔，可符合条带间平均重叠率的要求，但会导致水深较浅处出现漏测的情况，降低测量质量；而采取海水最浅处水深设计测线间隔，虽满足最浅处的重叠率的要求，却会导致水深较深处重叠过多，从而造成数据冗余，降低测量效率。

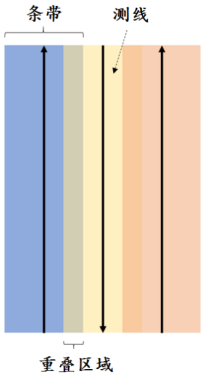

图 1　条带、测线及重叠区域

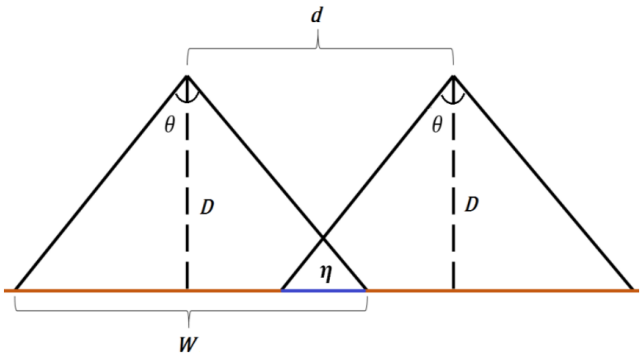

图 2　覆盖宽度、测线间距和重叠率之间的关系

1.2 问题重述

基于上述背景，请完成以下问题的求解：

问题 1：假设多波束换能器的开角为 120°，与测线方向垂直的平面与海底坡面之间的夹角（下称坡度）为 1.5°，海域中心处的海水深度为 70m。请建立多波束测深的覆盖宽度及相邻条带之间的重叠率的数学模型，计算不同测线距中心点处的距离下的海水深度、覆盖宽度以及与前一条测线的重叠率，结果保存为 *result*1.*xlsx* 文件。

问题 2：现给定一个矩形待测海域，测线方向与海底坡面的法向量在水平面上的投影夹角为 ，假设多波束换能器的开角为 120°，坡度为 1.5°，海域中心处的海水深度为

120m。请建立多波束测深覆盖宽度的数学模型，计算不同测量船距海域中心点距离、测线方向夹角下的覆盖深度，结果保存为 result2.xlsx 文件。

问题 3：现给定一个南北长 2 海里，东西宽 4 海里的矩形海域内，西深东浅，坡度为 1.5°，海域中心处的海水深度为 110m，多波束换能器的开角为 120°。在满足相邻条带之间的重叠率为 10% ~20% 且完全覆盖整个海域的要求下，设计一条测量长度最短的测线。

问题 4：根据提供的海水深度数据，在南北长 5 海里、东西宽 4 海里的某海域中设计满足如下要求的测线：

(1) 沿测线扫描形成的条带尽可能地覆盖整个待测海域；

(2) 相邻条带之间的重叠率尽量控制在 20% 以下；

(3) 测线的总长度尽可能短。

在设计出具体的测线后，请计算测线的总长度、漏测海区占总待测海域面积的百分比、在重叠区域中重叠率超过 20% 部分的总长度等 3 项指标。

二、问题分析

2.1 问题 1 的分析

由于本问题的测线平行于海底的坡面，故同一条测线的不同位置的多波束测深的覆盖宽度及相邻条带之间重叠率相同。我们可以借助正弦定理及相似三角形关系，通过测线距中心点处的距离得到该测线处的海水深度和覆盖宽度。根据测线相互平行且海底地形平坦时给出的重叠率定义，与前一条测线的重叠率为：

$$\eta' = 1 - \frac{\text{与前一条测线的间距}}{\text{该测线的覆盖宽度}} \tag{1}$$

代入与前一条测线的间距和该测线的覆盖宽度即可计算每条测线与前一条测线的重叠率。

2.2 问题 2 的分析

相较于问题 1，本问的测线不再平行于海底的坡面，但可以通过对测线方向进行平行于水平面的正交分解，得到与问题 1 中一样和海底坡面平行的方向的位移及与之垂直的方向的位移。而前者在测线方向一定并不会改变多波束测深的覆盖宽度，只有后者会有影响。此外，测线与海底坡面法向的夹角可通过空间立体几何的方法转化到覆盖区域截面与水平面的夹角上去，此时我们即可将本问化归到问题 1 中的情况。

2.3 问题 3 的分析

借助问题 2 的结果，我们可以得到满足相邻条带之间的重叠率满足要求时条带间距需要满足的约束，以完全覆盖整个待测海域的为绝对约束，测量长度尽可能短即测线间距尽可能大为优化目标，同时考虑两个不同测线间的重叠率变化及不同测线间夹角的变化对优化目标的影响。

2.4 问题 4 的分析

根据题意，本问需考虑三个优化目标，而三个目标互相制约且海域深度分布不均匀，可考虑将海域分割成多个小块，研究每个小块时可将其近似当作斜面。借助贪心的策略设计每条测线在每个小块中行进的方向，最后通过点标记等方法计算要求的 3 个指标。

三、模型假设

(1) 海水介质均匀，不影响多波束测深过程中声波的传播；

(2) 若地形平坦且测线相互平行，重叠率定义为 $\eta = 1 - \frac{d}{W}$，W 为条带的覆盖宽度，d 为相邻两条测线的间距；

(3) 若地形平坦但相邻条带不平行时，该测线上任意一点的重叠率重叠率定义为 $\eta = 1 - \frac{d}{W}$：W 为条带的覆盖宽度，其中 d 为该测线上某点到相邻测线的距离；

(4) 多波束测深仪器的换能器开角保持稳定，即每个声波束的发射角度保持不变；

(5) 假设海平面始终保持水平，即测线与水平面平行；

(6) 测线为直线或折线。

四、符号说明

符号	含义	单位
θ	多波束换能器的开角	°
α	坡度	°
β	测线方向与海底坡面的法向在水平面上投影的夹角	°
γ	每点形成的多波束测深条带与坡面的交线与水平面的夹角	°
d	平行测线的间距	m
$\tilde{d}$	测线距中心点处的距离	m

$\hat{d}$	测线上任意一点与相邻测线的垂距离	m
D	海域中心处的海水深度	m
η	重叠率	%
W	覆盖宽度	m

五、模型的建立与求解

5.1 问题 1 的模型建立与求解

5.1.1 模型建立

根据分析，考虑如图 3 所示的示意图：O 为中心点处测线的侧视点，AB 为此处多波束测深条带的覆盖区域在坡面上的覆盖宽度；O' 为距中心点处的距离为 $d_0(m)$ 的的测线的侧视点（此处规定向右是 d_0 的正方向，图 3 中的 d_0 为负值, 负号代表方向），$A'B'$ 为此处多波束测深条带的覆盖区域在坡面上的覆盖宽度；M, M' 分别为 O, O' 垂直于水平面且在海底坡面的投影点。图示中 OM 的长度即该点的海水深度 $D = 70m$，$O'M'$ 的长度即距中心点处的距离为 d_0 处的海水深度 D'。

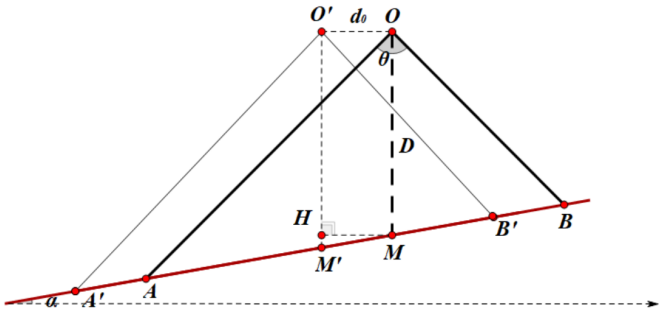

图 3　问题 1 的示意图

首先考虑计算 $W = |AB|\cos\alpha$：在 $\triangle OMA$ 中，由正弦定理可知：

$$\frac{|AM|}{\sin\frac{\theta}{2}} = \frac{|OM|}{\sin\angle OAB} \Rightarrow |AM| = D\cdot\frac{\sin\frac{\theta}{2}}{\sin\angle OAB}$$

同理考虑 $\triangle OBM$ 有 $|BM| = D\cdot\dfrac{\sin\frac{\theta}{2}}{\sin\angle OBA}$，由角度关系可知 $\angle OAB = \frac{\pi-\theta}{2}-\alpha, \angle OBA = \frac{\pi-\theta}{2}+\alpha$，所以:

$$\begin{aligned} W &= |AB|\cos\alpha = (|AM| + |BM|)\cos\alpha \\ &= D\cdot\left(\frac{\sin\frac{\theta}{2}}{\sin\left(\frac{\pi-\theta}{2}-\alpha\right)} + \frac{\sin\frac{\theta}{2}}{\sin\left(\frac{\pi-\theta}{2}+\alpha\right)}\right)\cos\alpha = \frac{D\sin\theta\cos^2\alpha}{\cos\left(\frac{\theta}{2}+\alpha\right)\cos\left(\frac{\theta}{2}-\alpha\right)} \end{aligned} \tag{2}$$

在直角梯形 $OMM'O'$ 中，可计算得：

$$D' = |O'M'| = |O'H| + |HM'| = D - d_0 \cdot \tan\alpha \tag{3}$$

由于 $\triangle OAB \sim \triangle OA'B'$，有：

$$\frac{D}{|AB|} = \frac{D'}{|A'B'|} \Rightarrow |A'B'| = D' \cdot \frac{|AB|}{D}$$

进一步有：

$$W' = |A'B'|\cos\alpha = \frac{D'\sin\theta\cos^2\alpha}{\cos\left(\frac{\theta}{2}+\alpha\right)\cos\left(\frac{\theta}{2}-\alpha\right)} = \frac{(D - d_0\cdot\tan\alpha)\sin\theta\cos^2\alpha}{\cos\left(\frac{\theta}{2}+\alpha\right)\cos\left(\frac{\theta}{2}-\alpha\right)} \tag{4}$$

根据前面分析中的式 (1)，本问中“与前一条测线的重叠率”为：

$$\eta' = 1 - \frac{d}{W'} \tag{5}$$

5.1.2 问题 1 的求解

代入 $\alpha = 1.5°$，$\theta = 120°$，$d = 200m$，$D = 70m$ 到式 (3) - (5) 中可得：

$$\begin{cases} D' = 70 - d_0 \cdot \tan 1.5° \\ W' = \dfrac{(70 - d_0\cdot\tan 1.5°)\sin 120°\cos^2 1.5°}{\cos 61.5°\cos 58.5°} \\ \eta' = 1 - \dfrac{200}{W'} \end{cases}$$

可以通过 python 编程计算出所需的指标值如下所示：

表 1　问题 1 的计算结果

测线距中心点处的距离/m	-800	-600	-400	-200	0	200	400	600	800
海水深度/m	90.95	85.71	80.47	75.24	70	64.76	59.53	54.29	49.05
覆盖宽度/m	315.71	297.53	279.35	261.17	242.99	224.81	206.63	188.45	170.27
与前一条测线的重叠率/%	—	32.78	28.40	23.42	17.69	11.03	3.21	-6.13	-17.46

上述结果均保留了两位小数，更高精度的结果详见 result1.xlsx 文件。

5.2 问题 2 的模型建立与求解

5.2.1 模型建立

根据分析，考虑如图 4 所示的示意图：O 为海域中心点，O' 为测量船的位置，则 $|OO'|$ 为测量船距海域中心点处的距离 $\tilde{d}$（海里），且 AB 为 O 处多波束测深条带的覆盖区域。为方便处理，考虑以点为原点，建立如图 4 所示的空间直角坐标系。

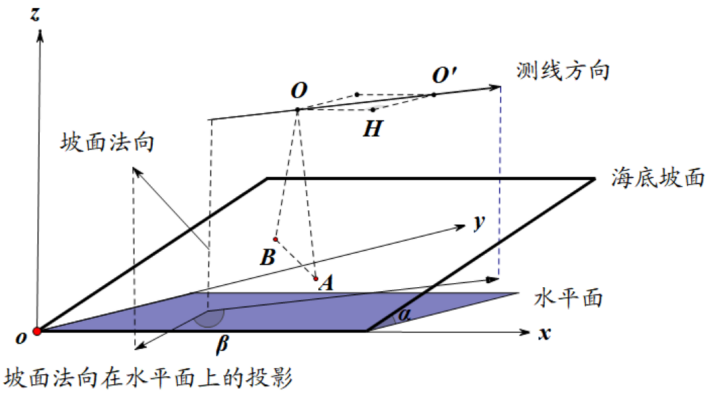

图 4　问题 2 的示意图

则由题意可知，测线的方向向量 $\vec{v}$、坡面的法向量 $\vec{n}$ 以及水平面的法向量 $\vec{n}_0$ 分别为：

$$\vec{v} = (\sin\beta, -\cos\beta, 0)$$

$$\vec{n} = (0, -\sin\alpha, \cos\alpha)$$

$$\vec{n}_0 = (0, 0, 1)$$

对于 $\overrightarrow{AB}$ 的方向向量 $\vec{u}$，由 AB 是过点 O 且与 $\vec{v}$ 垂直的平面和海底坡面的交线可知：

$$\begin{cases} \vec{u}\cdot\vec{v} = 0 \\ \vec{u}\cdot\vec{n} = 0 \end{cases} \quad \Rightarrow \quad \vec{u} = (\cos\beta, \sin\beta, \sin\beta\tan\alpha)$$

此时可知 AB 与水平面的夹角 γ 满足：

$$\sin\gamma = \frac{\vec{u}\cdot\vec{n}_0}{|\vec{u}||\vec{n}_0|} = \frac{\sin\beta\tan\alpha}{\sqrt{\cos^2\beta + \sin^2\beta + \sin^2\beta\tan^2\alpha}} = \frac{\sin\beta\tan\alpha}{\sqrt{1 + \sin^2\beta\tan^2\alpha}}$$

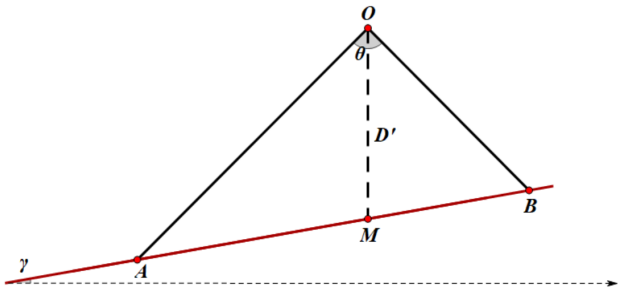

图 5　问题 2 的辅助说明图

进一步地，可化归为问题 1 中的情况（如图 5 所示），结合式 (2) 可知：

$$W = \frac{D\sin\theta\cos^2\gamma}{\cos\left(\frac{\theta}{2}+\gamma\right)\cos\left(\frac{\theta}{2}-\gamma\right)}, \quad \text{其中}\gamma = \arcsin\left(\frac{\sin\beta\tan\alpha}{\sqrt{1+\sin^2\beta\tan^2\alpha}}\right) \tag{6}$$

对于测量船不在海域中心的情况，考虑对向量 $\overrightarrow{OO'}$ 作如图 4 所示的平行于水平面的正交分解，即：平面 $OO'H$ // 水平面，$\overrightarrow{OH} \perp \overrightarrow{O'H}$ 且 $\overrightarrow{OH}$ // 海底坡面。

根据问题 1 的分析，位移 $\overrightarrow{OH}$ 并不会改变测量船的覆盖宽度，而位移 $\overrightarrow{O'H}$ 会对测量船的覆盖宽度产生影响，且其影响与式 (4) 中的 d 的改变相同。由题意可知：

$$|HO'| = \tilde{d}\cos(\pi - \beta) = -\tilde{d}\cos\beta$$

此处 HO' 的正负并不表示大小，而表示行进方向是否朝着更浅处。结合式 (6) 可知：

$$W' = \frac{(D + \tilde{d}\cos\beta \cdot \tan\alpha)\sin\theta\cos^2\gamma}{\cos\left(\frac{\theta}{2} + \gamma\right)\cos\left(\frac{\theta}{2} - \gamma\right)} = \frac{(D + \tilde{d}\cos\beta \cdot \tan\alpha)\sin\theta(1 - \sin^2\gamma)}{\cos^2(\frac{\theta}{2}) - \sin^2\gamma} \tag{7}$$

其中 $\gamma = \arcsin\left(\dfrac{\sin\beta\tan\alpha}{\sqrt{1 + \sin^2\beta\tan^2\alpha}}\right)$

5.2.2 问题 2 的求解

代入 $\alpha = 1.5°$，$\theta = 120°$，$D = 120m$ 到式 (7) 中可得：

$$W' = \frac{(120 + \tilde{d}\cos\beta \cdot \tan 1.5°)\sin 120°(1 - \sin^2\gamma)}{\frac{1}{4} - \sin^2\gamma}, \quad \text{其中}\gamma = \arcsin\left(\frac{\sin\beta\tan 1.5°}{\sqrt{1 + \sin^2\beta\tan^2 1.5°}}\right)$$

可通过 python 编程计算出要求的覆盖宽度结果如下（下述结果均保留了两位小数，更高精度的结果详见 result2.xlsx 文件）：

表 2　问题 2 的计算结果

覆盖宽度/m		测量船距海域中心点处的距离/海里							
		0	0.3	0.6	0.9	1.2	1.5	1.8	2.1
测线方向夹角/°	0	415.69	466.09	516.49	566.89	617.29	667.69	718.09	768.48
	45	416.12	451.79	487.47	523.14	558.82	594.49	630.16	665.84
	90	416.55	416.55	416.55	416.55	416.55	416.55	416.55	416.55
	135	416.12	380.45	344.77	309.10	273.42	237.75	202.08	166.40
	180	415.69	365.29	314.89	264.50	214.10	163.70	113.30	62.90
	225	416.12	380.45	344.77	309.10	273.42	237.75	202.08	166.40
	270	416.55	416.55	416.55	416.55	416.55	416.55	416.55	416.55
	315	416.12	451.79	487.47	523.14	558.82	594.49	630.16	665.84

5.3 问题 3 的模型建立与求解

5.3.1 模型建立

按照分析，该海域俯视图如图 6 所示，考虑图示相邻两条测线：PQ, P_1Q_1, P_2Q_2 为三条垂直于 OP_2 的线段，两测线与海底坡面的法向在水平面上投影的夹角分别为 β_1, β_2，X 表示该海域的南北长 2 海里，Y 则表示东西宽 4 海里。根据式 (5) 和式 (7) 可知：

$$\eta = 1 - k \cdot \frac{\hat{d}}{\hat{D}}, \quad \text{其中} k(\beta) = \frac{\cos(\frac{\theta}{2} + \gamma)\cos(\frac{\theta}{2} - \gamma)}{\sin\theta\cos^2\gamma}, \gamma = \arcsin\left(\frac{\sin\beta\tan\alpha}{\sqrt{1 + \sin^2\beta\tan^2\alpha}}\right) \tag{8}$$

其中 $\hat{d}$ 表示 P 点所在测线与相邻测线的垂直距离 $|PQ|$[1]，$\hat{D}$ 表示位置 O 的海水深度。

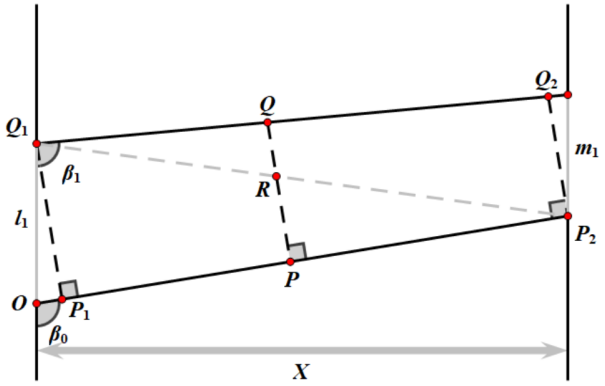

图 6　问题 3 的示意图

我们记 $|Q_1Q| = t > 0, |OO'| = l_1$，点 O 处深度为 $\hat{D}_0$，则：

$$D_q = \hat{D}_0 - l_1\tan\alpha - t\cos(\pi - \beta_0)\tan\alpha = \hat{D}_0 - l_1\tan\alpha + t\cos\beta_0\tan\alpha$$

考虑等腰梯形 PQQ_1P_1，可知：

$$\begin{aligned}\hat{d} &= |QP| = |Q_1P_1| - |Q_1Q|\cos\angle QQ_1P_1 = l_1\cos\angle OQ_1P_1 - t\cos\angle QQ_1P_1 \\ &= l_1\cos\left(\beta_0 - \frac{\pi}{2}\right) - t\cos\left(\beta_1 - \beta_0 + \frac{\pi}{2}\right) = l_1\sin\beta_0 + t\sin(\beta_1 - \beta_0)\end{aligned}$$

结合式 (8) 和上述两式可知：

$$\eta_q = 1 - k \cdot \frac{\hat{d}}{D_q} = 1 - k \cdot \frac{l_1\sin\beta_0 + t\sin(\beta_1 - \beta_0)}{\hat{D}_0 - l_1\tan\alpha + t\cos\beta_0\tan\alpha} \tag{9}$$

此处 η_q 关于 l_1 单调递减。不难发现，如果希望总测线长度尽可能短，那么测线总数需要尽可能少，换言之，这里的 l_1 就需要尽可能的大。而某条测线与前一条测线的重叠率将小于其与后一条测线的重叠率，就优化目标而言，后者并没有考虑的意义。

此外，如图 6 中情况，t 的取值范围为 $[0,|Q_1Q_2|]$，对右上角的三角形中使用正弦定理可知：

$$\frac{X\csc\beta_1-|Q_1Q_2|}{\sin\left(\beta_0-\frac{\pi}{2}\right)}=\frac{m_1}{\sin\left(\frac{\pi}{2}-\beta_0+\beta_1\right)}\Rightarrow|Q_1Q_2|=X\csc\beta_1-\frac{m_1\cos\beta_0}{\cos(\beta_0-\beta_1)}$$

同理对于一般情况，我们不难得到如下结论：$t\in[tmin,tmax]$，其中：

$$tmin_i=\max\left\{0,\frac{l_i\cos\beta_{i-1}}{\cos(\beta_{i-1}-\beta_i)}\right\},tmax_i=\frac{X}{\sin\beta_{i-1}}+\min\left\{0,\frac{m_i\cos\beta_{i-1}}{\cos(\beta_{i-1}-\beta_i)}\right\}$$

而对于第一条测线，其只需要满足图 3 中的 $|MA|\geq l_1$，即：

$$\frac{(\hat{D}_1-l_1\tan\alpha)\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)}\geq l_1\sec\alpha \tag{10}$$

设 n 条测线与海底坡面的法向在水平面上投影的夹角分别为 $\beta_1,\beta_2,\cdots,\beta_n$，则所有测线的长度和为：

$$\sum_{i=1}^{n}\frac{X}{\sin(\pi-\beta_i)}\quad\text{i.e.}\quad\sum_{i=1}^{n}\frac{X}{\sin\beta_i} \tag{11}$$

综合式 (9) - (11) 可得优化问题如下：

$$\min\sum_{i=1}^{n}\frac{X}{\sin\beta_i}$$

$$\text{s.t.}\begin{cases}\dfrac{(\hat{D}_1-l_1\tan\alpha)\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)}\geq l_1\sec\alpha\\ \eta_i(t)=1-k\cdot\dfrac{l_i\sin\beta_{i-1}+t\sin(\beta_i-\beta_{i-1})}{\hat{D}_{i-1}-(l_i-t\cos\beta_i)\tan\alpha},\quad i=2,\cdots,n,\\ \hat{D}_1=110+2*1852\tan\alpha,\hat{D}_i=\hat{D}_1-\left(\sum_{j=1}^{i-1}l_j\right)\cdot\tan\alpha,\quad i=2,\cdots,n\\ 10\%\leq\eta_i(t)\leq 20\%,\quad\forall t\in[tmin_i,tmax_i],i=2,3,\cdots,n\\ \quad\text{其中}tmin_i=\max\left\{0,\dfrac{l_i\cos\beta_{i-1}}{\cos(\beta_{i-1}-\beta_i)}\right\},tmax_i=\dfrac{X}{\sin\beta_{i-1}}+\min\left\{0,\dfrac{m_i\cos\beta_{i-1}}{\cos(\beta_{i-1}-\beta_i)}\right\}\\ m_i+X\cot\beta_i=l_i+X\cot\beta_{i-1},\quad i=1,2,\cdots,n\\ \sum_{j=1}^{n}l_j\geq Y>\sum_{j=1}^{n-1}l_j\\ \beta_i\in(0,180^\circ),\quad i=1,2,\cdots,n\\ l_i>0,m_i>0,\quad i=1,2,\cdots,n\end{cases} \tag{12}$$

其中 n 需要提前取定，若存在可行解则说明这个条数是可行的，同时需要尝试不同取值的 n 以找到全局的最优解。

5.3.2 问题 3 的求解

现考虑上述求解思路中的特殊情况，$\beta_1 = \beta_2 = \cdots = \beta_n$，对该规划进行初步求解，现考虑第一组相邻测线和最后一组相邻测线，可知当 β 角偏离 90° 过大时，相邻测线形成的覆盖区域在两条边界线上的区间范围将会出现没有交集的情况，即该测线组不同覆盖整个海域，记 $\varepsilon = \beta - 90°$, 调整 python 程序中的 β 值，可得 ε 的大致范围为 [89°, 91°]。

首先考虑最特殊的情况，即 $\beta_1 = \beta_2 = \cdots = \beta_n = 90°$，此时只需要求解最少的测线条数即可。为此，可设计如下算法 1：

Algorithm 1 问题 3 特殊情况求解算法

Input: $X = 2\,(\text{海里}), Y = 4\,(\text{海里}), \theta = 120°, \alpha = 1.5°, D_{\text{中心}} = 110m, \beta_i = 90°, i = 0, \cdots, n$

Output: 最短测量长度

(1) 初始化：$i \leftarrow 1, \hat{D}_0 = 110 + 2 * 1852 \tan\alpha, \hat{D}_{\text{last}} = 110 - 2 * 1852 \tan\alpha$;

(2) 计算 $k = \dfrac{\cos(\frac{\theta}{2} + \alpha)\cos(\frac{\theta}{2} - \alpha)}{\sin\theta \cos^2\alpha}$;

(3) 根据下述约束可得 l_1 取值范围，令 l_1^* 取该范围中最大的 l_1, 令 l_m^* 取该范围中最大的 l_m, 即两个边界处满足最优：

$$\frac{(\hat{D}_0 - l_1 \tan\alpha)\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2} + \alpha\right)} \geq \frac{l_1}{\cos\alpha}, \quad \frac{(\hat{D}_m + l_1 \tan\alpha)\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2} - \alpha\right)} \geq \frac{l_1}{\cos\alpha}$$

(4) $\hat{D}_1 = \hat{D}_0 - l_1^* \tan\alpha, \hat{D}_m = \hat{D}_0 - l_1^* \tan\alpha$;

(5) **While** $\hat{D}_i > 110 - 2 * 1852 \tan\alpha$ **do**

令 l_i* 为满足下述约束的最大的 l_i

$$\eta_i = 1 - \frac{kl_i}{\hat{D}_i} \in [0.1, 0.2]$$

令 $\hat{D}_{i+1} = \hat{D}_i - l_{i+1}^* \tan\alpha, i \leftarrow i + 1$;

End

(6) **return** 最短测量长度 iX.

根据上述算法编写 Matlab 程序可得结果：共 37 条平行测线，测量长度为 $137048m$，具体分布情况如下图 7 所示

进一步通过遗传算法求解 $\beta_1, \beta_2, \cdots, \beta_n$ 不相等的情况。根据式 (12) 中的模型，考虑设计遗传算法的求解方案如下：

[1]. **染色体的编码**：

由于目标函数中的变量为 $\beta_1, \beta_2, \cdots, \beta_n$，所以考虑将 β_i 体现在染色体的编码中去，由于角度为连续变量，我们考虑将其离散化，即建立从 0-1 编码到角的一一对应：

$$x_1 x_2 \cdots x_m, \text{ 其中} x_i \in \{0, 1\}, i = 1, 2, \cdots, m \quad \leftrightarrow \quad \beta = \frac{\pi}{2^m - 1} \cdot \left(\sum_{i=1}^{m} x_i \times 2^{m-i}\right)$$

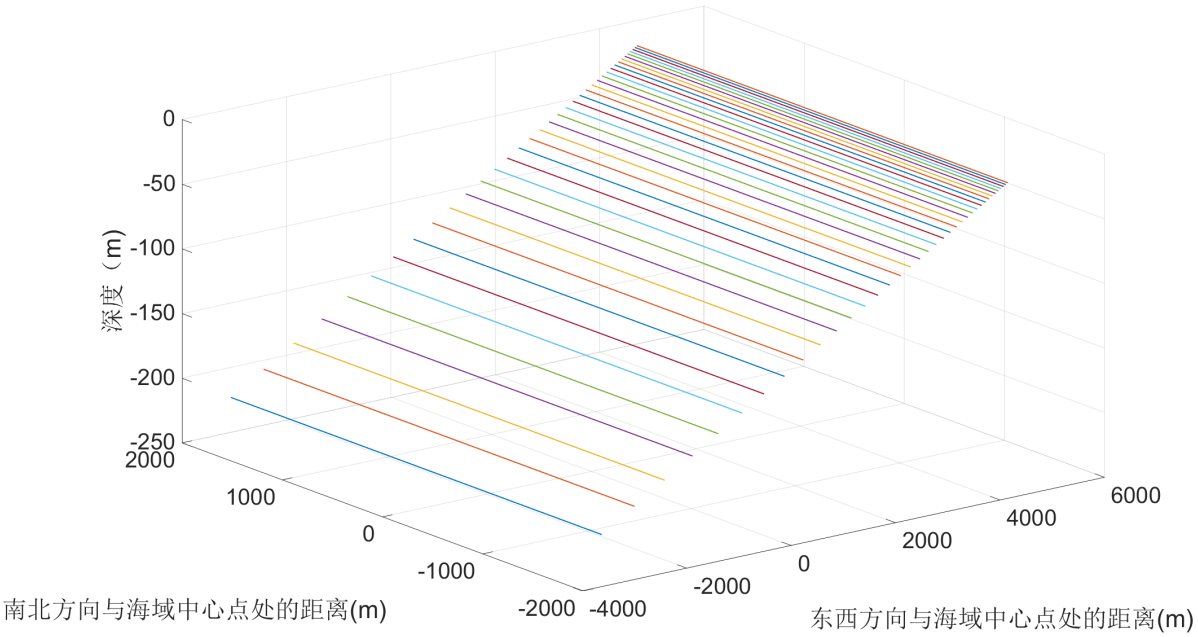

图 7　问题 3：测线南北走向且相互平行情况的求解结果

不难发现，m 越大，相应的角 β 的值的精度将越高，即越接近于连续变量。基于这个，我们可得种群的矩阵表示如下：

$$H=\begin{pmatrix} x_{11}^{(1)} & \cdots & x_{1m}^{(1)} & x_{21}^{(1)} & \cdots & x_{2m}^{(1)} & \cdots & x_{n1}^{(1)} & \cdots & x_{nm}^{(1)} \\ x_{11}^{(2)} & \cdots & x_{1m}^{(2)} & x_{21}^{(2)} & \cdots & x_{2m}^{(2)} & \cdots & x_{n1}^{(2)} & \cdots & x_{nm}^{(2)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{11}^{(N)} & \cdots & x_{1m}^{(N)} & x_{21}^{(N)} & \cdots & x_{2m}^{(N)} & \cdots & x_{n1}^{(N)} & \cdots & x_{nm}^{(N)} \end{pmatrix}$$

其中

- H 的每一行表示一个个体，每个个体包含 n 段长度为 m 的基因片段，每段基因片段表示一个 β_j.
- $x_{ij}^{(k)} \in \{0,1\}, i=1,2,\cdots,n, j=1,2,\cdots,m, k=1,2,\cdots,N.$

根据特殊情况的求解结果，取 $m=10, n=40.$

[2]. **个体评估与检验**：

得到了种群的矩阵后，在进行选择复制等选择之前需先对染色体进行适应度评估，即根据适应度的大小进行排序，得到优质个体。考虑本问中的适应度函数，设由染色体 $(x_{i1}, x_{i2}, \cdots, x_{im})$ 所确认的角为 β_i，

考虑在 $\beta_1, \beta_2, \cdots, \beta_n$ 取定的情况下对式 (12) 中的约束进行检验，寻找使得 $\hat{D}_{n'} \geq Y > \hat{D}_{n'-1}$ 的最小正整数 n'（n' 不一定等于 n）。若 $n' \leq n$，那么此时相应的目标函

数值为

$$fval = \sum_{i=1}^{n'} \frac{X}{\sin\beta_i}$$

由于后续将考虑轮盘赌的方法进行选择复制，所以定义适应度函数为其调整完以后的距离偏差量的倒数

$$fitness = \frac{1}{fval} = 1\Big/\left(\sum_{i=1}^{n'} \frac{X}{\sin\beta_i}\right)$$

若 $\hat{D}_n < Y$，说明此时该个体无法对应一个符合约束的方案，定义此时的 $fval = \infty, fitness = \varepsilon$，其中 ε 是一个足够小的正数。

[3]. **选择、交叉与变异**：

对每一条染色体进行评估与检验之后，即可进行选择复制，在此我们考虑采用轮盘赌选择法，即各个个体的选择概率与其适应度成正比，之后按照一定的概率进行交叉和变异。

对于交叉，考虑使用单点交叉的方法，即随机抽取两个个体，在个体串中随机设定一个交叉点，在实行交叉时，该点前或后的两个个体的部分结构进行互换，生成两个新个体。对于变异，考虑针对种群矩阵中的基因进行单点突变的方式。

此外我们设定在达到最大代数或者最优个体的适应度不再上升后，就认为已寻得近似的最优解。

根据上述方案编写 python 程序，设定遗传算法的相关参数：种群大小 $N = 20$，最大代数 IterMax= 100，交叉率 $p_e = 0.5$，变异率 $p_m = 0.1$.

表 3　遗传算法求解结果（夹角 β_i）

角编号	角度 (°)	角编号	角度 (°)	角编号	角度 (°)	角编号	角度 (°)
β_1	89.47265625	β_9	88.9453125	β_{17}	89.12109375	β_{25}	88.9453125
β_2	88.9453125	β_{10}	88.76953125	β_{18}	90.0	β_{26}	88.41796875
β_3	88.59375	β_{11}	89.296875	β_{19}	89.6484375	β_{27}	89.47265625
β_4	88.9453125	β_{12}	89.82421875	β_{20}	88.06640625	β_{28}	88.59375
β_5	90.0	β_{13}	88.41796875	β_{21}	89.6484375	β_{29}	89.47265625
β_6	88.59375	β_{14}	89.6484375	β_{22}	89.47265625	β_{30}	90.0
β_7	88.2421875	β_{15}	89.296875	β_{23}	90.0	β_{31}	89.47265625
β_8	88.59375	β_{16}	89.296875	β_{24}	89.6484375		

可得求解结果如下：共 31 条直测线，测量长度为 114824m，此时各直测线与坡面法向在水平面上的投影的夹角 β 见上表 3。具体优化过程的最优适应度函数值如图 8 所示：

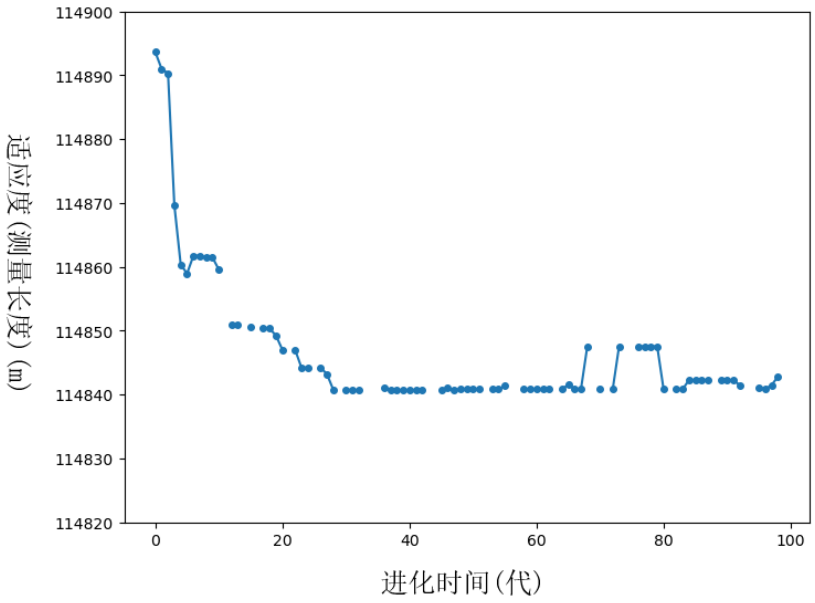

图 8　问题 3：遗传算法适应度的优化过程

5.4 问题 4 的模型建立与求解

5.4.1 模型建立

通过附件中的数据，我们可以绘制待测海域的深度分布图如下所示：

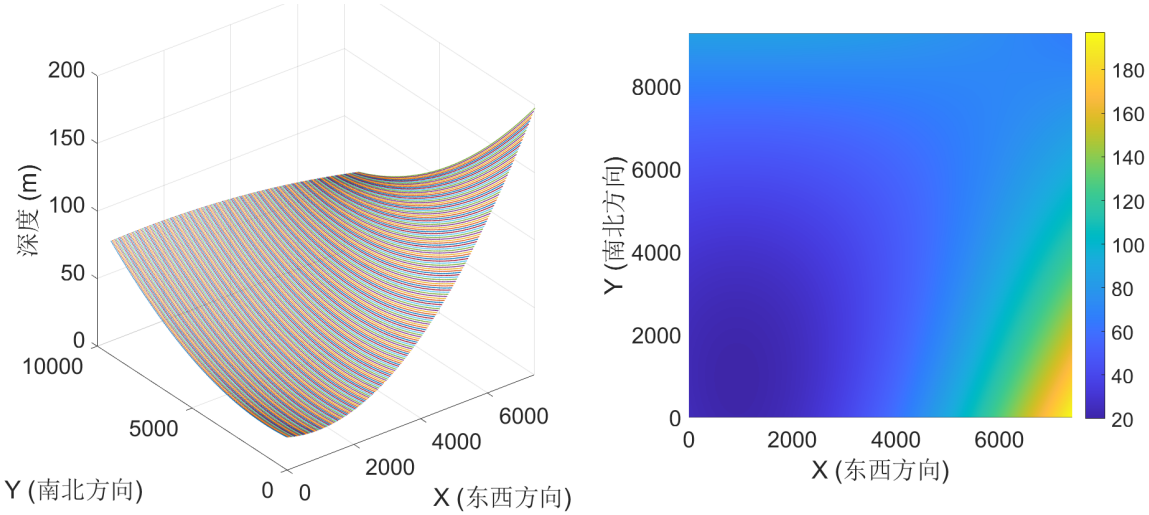

图 9　待测海域深度展示图

观察图 9，可以发现海底深度分布较为平滑，所以考虑将其分割成一些小块进行处理，在各个小块区域将地形近似看成斜坡，进行测线设计。在局部决策时，为了使得测线的总长度尽可能短，测线与边界（南北或东西）的夹角尽可能要接近于 $\frac{\pi}{2}$。但如果要尽可能覆盖待测海域或者重叠率不高于 20%，行进方向可能会改变较大，考虑到实际情况中测量船的方向转变不会太大[1]，以及“测线长度尽可能小”的要求，满足要求 (1)(2)

的行进方向若与南北（或东西）方向的夹角与 $\frac{\pi}{2}$ 相差较大，考虑选择一个与 $\frac{\pi}{2}$ 较为接近的值代替，通过漏测部分海域或部分区域重叠率过大适当减少测线长度。

综合上述分析，我们可以设计基于贪心策略的测线设计算法如下图 10 所示。

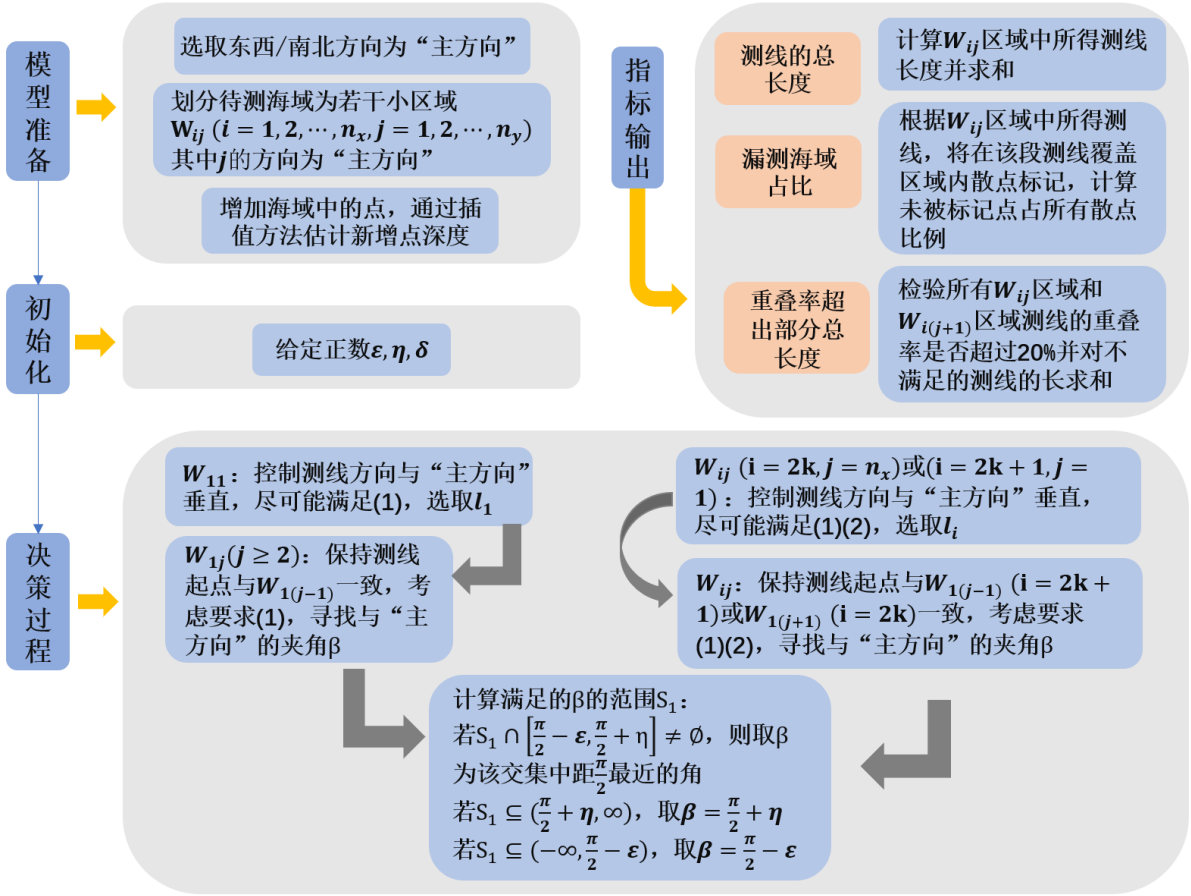

图 10　问题 4 求解算法流程图

在上述算法中，考虑矩形分块区域边界处，在南北方向的相邻区域拼接处，考虑实际情况，多波束测量船的测线应为连续，故使得不同区域的测线顺次相接。对于东西方向相邻区域的拼接处，从西向东视角的侧视图如下图所示：

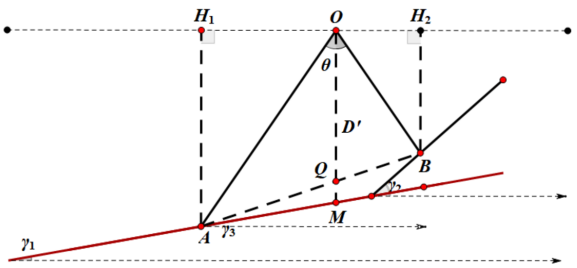

图 11　问题 4 从西向东侧视图

在此情况下，由问题 2 的结论可知，想计算 O 点的重叠率，只需求图中 $|OQ|$ 的长

度即可。

$$W' = \frac{|OQ|\sin\theta(1-\sin^2\gamma_3)}{\cos^2(\frac{\theta}{2})-\sin^2(\gamma_3)}$$

$|OQ|$ 的长度可通过问题 2 中所建立的空间直角坐标系进行，根据 $\overrightarrow{OA}$ 垂直于测线方向向量 $\vec{v}$ 且与水平面的法向量夹角为 60°, 可求解的 $\overrightarrow{OA} = (\cos\beta, -\sin\beta, \frac{\sqrt{3}}{3})$，$A$ 为坡度为 γ_1 的斜面与直线 OA 的交点，联立方程进一步求得 A 点坐标, 同理可求得 B 点坐标，进一步求得 γ_3，代入可写进以上重叠率表达式。

在算法设计求解时，选取适合表达式进行覆盖宽度的计算。对于相邻测线重叠率的选取，我们给定 15-25% 的区间，对于区间进行离散化处理，根据上一步结果选取最为合适的重叠率，检查满足相邻区域最优的局部解是否为全局最优解，若不是则返回计算。三个优化目标的计算方式如图 10所示，测线长度直接进行求和计算，重叠率超出部分进行检验后进行求和计算，漏测海域占比考虑通过仿真标记散点计算未标记散点占所有散点的比例，我们赋予漏测海域面积的优化目标最高的优先因子进行求解。

5.4.2 问题 4 的求解

根据图 10 所示的算法用 python 进行求解，选取南北方向为“主方向”，可得测线总长度为 442371.69m，漏测面积占比为 2.13%，重叠率超过 20% 的部分长度为 4527m，同时测线情况和未覆盖区域（白色的为未覆盖区域）如下图 12 所示：

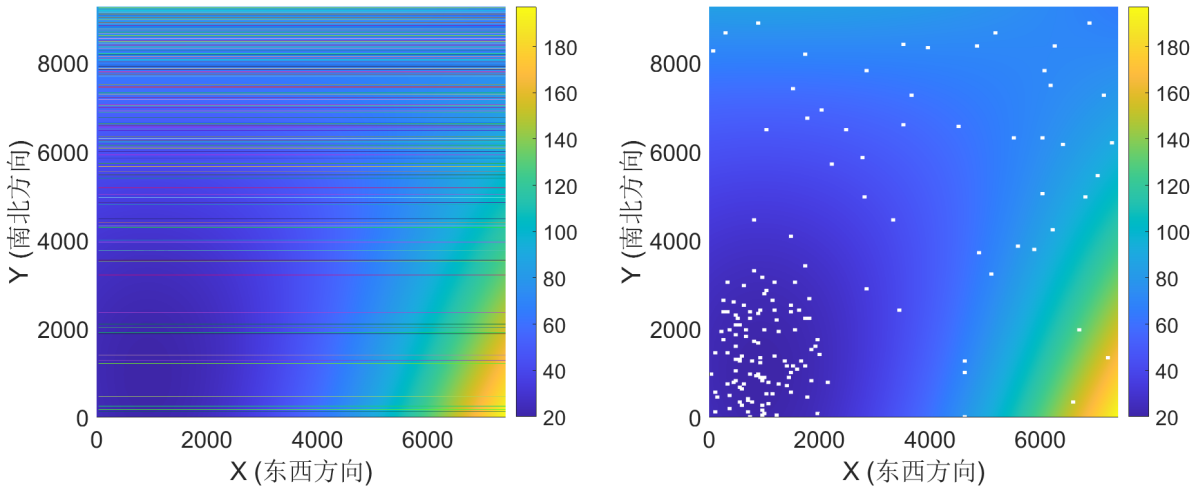

图 12　问题 4 求解结果

六、模型评价

6.1 模型的优点

(1) 问题 1、2 的模型清晰、简洁、科学，能够很好地刻画多线束测深过程任意一条测线上不同点的覆盖宽度以及相邻条带之间的重叠率。

(2) 问题 3 的模型从实际出发，充分利用了前面的模型，将问题转化为非线性规划模型[2]，为遗传算法[3]的设计提供了理论依据。

(3) 基于贪心策略的问题 4 的模型，能够适用于不平坦地形情况的测线路径选择问题，模型具有较好的可迁移性。

6.2 模型的缺点

(1) 启发式算法较难获得全局的最优解，只能在一定程度上给出问题的近似最优解。

(2) 对于地形不平坦的情况划分区域时相当于取了近似，与实际情况存在一定差异。

参考文献

[1] 中国地质调查局地质调查技术标准: 海洋多波束测量规程: DD 2021-01[S]. 北京: 自然资源部中国地质调查局, 2021: 7.

[2] 姜启源, 谢金星, 叶俊等. 数学模型[M]. 北京: 高等教育出版社, 2011.

[3] 司守奎, 孙兆亮等. 数学建模算法与应用[M]. 2 版. 北京: 国防工业出版社, 2017.

A 附录

A.1 支撑材料文件列表

支撑材料	q1q2.py	
	situation1.m	
	q3ga.py	
	analysis1.m	
	q4.py	
	数据	附件.xlsx
		depth.xlsx
	结果	result1.xlsx
		result2.xlsx

B 源程序

B.1 问题 1、2 代码

```
import math
import numpy as np
distance0 =[-800,-600,-400,-200,0,200,400,600,800]
D=[]
W=[]
Overlap_rate=[]
for i in range(len(distance0)):
    Di=70-distance0[i]* math.tan(math.radians(1.5))
    Wi=(70-distance0[i]* math.tan(math.radians(1.5)))*math.sin(math.radians(120))*math.
        cos(math.radians(1.5))*math.cos(math.radians(1.5))/(math.cos(math.radians(61.5))*
        math.cos(math.radians(58.5)))
    D.append(Di)
    W.append(Wi)
print('海水深度: '+str(np.array(D)))
print('覆盖宽度:'+str(np.array(W)))
for i in range(1,9):
    Overlap_ratei=1-200/W[i]
    Overlap_rate.append(Overlap_ratei)
print('与前一条测线的重叠率:'+str(np.array(Overlap_rate)))
```

```
distant1   =[0,0.3,0.6,0.9,1.2,1.5,1.8,2.1]
angle=[0,45,90,135,180,225,270,315]
W_2=[]
distant1_convert=[element * 1852 for element in distant1]
for i in range(8):
    for j in range(8):
        singamai=math.sin(math.radians(angle[i]))*math.tan(math.radians(1.5))/math.sqrt
            (1+math.sin(math.radians(angle[i]))*math.sin(math.radians(angle[i]))*math.tan(
            math.radians(1.5))*math.tan(math.radians(1.5)))
        W_2i=(120+distant1_convert[j]*math.cos(math.radians(angle[i]))*math.tan(math.
            radians(1.5)))*math.sin(math.radians(120))*(1-singamai*singamai)/(0.25-
            singamai*singamai)
        W_2.append(W_2i)
arr=np.array(W_2)
reshaped_arr = arr.reshape((8,8))
print(reshaped_arr)
```

B.2 问题 3 代码

B.2.1 平行情况求解

```
% 输入参数
X = 2*1852;
Y = 2*1852;
theta = deg2rad(120);
alpha = deg2rad(1.5);
D_center = 110;

% 初始化
i = 1;
D_0 = D_center + 2 * 1852 * tan(alpha);
k = (cos(theta/2 + alpha) * cos(theta/2 - alpha)) / (sin(theta) * cos(alpha)^2);
L = [];
D = [];

% 计算第一个 l 的值
l_1 = sin(theta/2) * D_0 / (cos(theta/2 + alpha)/cos(alpha) + sin(theta/2)*tan(alpha));

```

```
% 计算第二个点到倒数第二个点的 l 和 D
D_1 = D_0 + l_1 * tan(alpha);
L = [L, l_1];
D = [D, D_1];
l = l_1;
d = D_1;

% 循环计算后续点的 l 和 D
while D(i) > D_center - 2 * 1852 * tan(alpha)
    l = (0.9*D(i)) / (0.9*tan(alpha) + k);
    d = D(i) - l * tan(alpha);
    L = [L, l];
    D = [D, d];
    i = i + 1;
end

% 返回最短测量长度
shortest_length = i * X;

% 计算需要条数
disp(i);

% 计算最短测量长度
disp(['最短测量长度: ', num2str(shortest_length)]);

Y_start = zeros(1, i)-1852;
Y_end = ones(1, i) * Y-1852;
X_start = cumsum(L)-X;
X_end = X_start;
Z_start =-D(1:i);
Z_end = Z_start;

% 创建图形对象
figure;
plot3([X_start; X_end], [Y_start; Y_end], [Z_start; Z_end], 'LineWidth', 0.5);

% 设置坐标轴标签
xlabel('东西方向与海域中心点处的距离(m)');
ylabel('南北方向与海域中心点处的距离(m)');
zlabel('深度 (m)');
```

```
58
59 % 设置坐标轴刻度
60 set(gca, 'TickDir', 'in');
61 set(gca, 'TickLength', [0.02, 0.02]);
62 set(gca, 'FontSize', 8);
63
64
65 grid on;
66
67 % 显示图形
```

B.2.2 遗传算法

```
1  #一些基础函数
2  #角度和二进制的转换
3  def two_to_ten(num):
4      sum = 0
5      num = str(num)
6      length = len(num)
7      for x in range(length):
8          sum += int(num[length-1-x]) * pow(2, x)
9      return(sum)
10 import math
11 #把 bit 值转为弧度制角
12 def twoBit_to_angle(individual):
13     angle_list = []
14     for i in range(int(len(str(individual))/10)):#len(str(individual))/10 算出个体数目
15         b = str(individual)
16         number = int(b[10*i:10*i+10])
17         angle = math.pi/1024*(two_to_ten(number)+1)#"'angle 角投影体系要变化"'
18         angle_list.append(angle)
19     return(angle_list)
20 #解 lj 的方程 f A*X²+B*X+C/（DX+E） g
21 from scipy.optimize import fsolve,root
22 import sympy as sp
23 import math
24 def k(beta):
25     g = math.asin(math.sin(beta)*math.tan(math.radians(1.5))/(math.sqrt(1+math.sin(beta)
           *math.sin(beta)*math.tan(math.radians(1.5))*math.tan(math.radians(1.5)))))
```

```
    k_beta = math.cos(math.radians(60)+g)*math.cos(math.radians(60)-g)/(math.sin(math.
        radians(120))*math.cos(g)*math.cos(g))
    return k_beta
def solve_lj_before(Dj,beta_j,beta_j_1):#判断与前一条的是否重合
    # 定义符号
    lj  = sp.symbols('lj')
    X = 2*1852
    alpha = math.radians(1.5)
    sin_beta_j = math.sin(beta_j)
    sin_beta_j_1 = math.sin(beta_j_1)
    cos_beta_j = math.cos(beta_j)
    cos_beta_j_1 = math.cos(beta_j_1)
    tan_beta_j = math.tan(beta_j)
    tan_beta_j_1 = math.tan(beta_j_1)
    tan_alpha = math.tan(alpha)
    cos_beta_delta = math.cos(beta_j_1-beta_j)
    sin_beta_delta = math.sin(beta_j-beta_j_1)
    mj = lj+X/tan_beta_j_1-X/tan_beta_j
    t1  = max(0,cos_beta_j_1/cos_beta_delta)*lj
    t2  = X/sin_beta_j_1+min(0,cos_beta_j_1/cos_beta_delta)*mj
    kj  = k(beta_j)
    f1  = kj*(lj*sin_beta_j_1+t1*sin_beta_delta)/(Dj-(lj-t1*cos_beta_j)*tan_alpha)
    f2  = kj*(lj*sin_beta_j_1+t1*sin_beta_delta)/(Dj-(lj-t1*cos_beta_j)*tan_alpha)
    x1 = sp.solve(f1<=0.9,lj)
    x2 = sp.solve(f2<=0.9,lj)
    return x1,x2
#lj 计算
def  lj_result (Dj,beta_j,beta_j_1):
    import re
    x1,x2 = solve_lj_before(Dj,beta_j,beta_j_1)
    strs  = str(x1)+str(x2)
    input_str = strs
    # 使用正则表达式匹配前面有'= ' 的数字模式
    pattern = r"= (\d*\.\d+|\d+)"
    # 使用 findall 函数找到匹配的数字
    matches = re.findall(pattern, input_str)
    # 将匹配的数字转换为浮点数列表
    numbers = [float(match) for match in matches]
    select_numbers = []
    for  i  in  numbers:#i 表示 lj 的最大值
```

```
            if  i>0 and i<4*1852:
                select_numbers.append(i)
        if  select_numbers == []:
            return  'void'
        else:
            lj  = min(select_numbers)
            return  lj
import math
#适应度计算
D1 = 110+2*1852*math.tan(math.radians(1.5))
def  fitness(individual):#输入的是 10bit*n
        angle_list  = twoBit_to_angle(individual)#拆分成角度
        angle_list  = [math.radians(90)] + angle_list#加入 beta0
        Dj = D1
        lj_list  = []
        sum_lj = 0
        X = 2*1852
        Y = 4*1852
        fitness  = float('inf')
        for  i  in  range(2,len(angle_list)):
            if  i  ==2:
                lj  = Dj*math.sin(math.radians(60))/(math.cos(math.radians(60+1.5))/math.cos(
                    math.radians(1.5))+math.tan(math.radians(1.5))*math.sin(math.radians(60))
                    )
                sum_lj+=lj
            else:
                beta_j = angle_list[i]
                beta_j_1 = angle_list[i-1]
                tan_beta_j = math.tan(beta_j)
                tan_beta_j_1 = math.tan(beta_j_1)
                lj  = lj_result(Dj,beta_j,beta_j_1)
                if  lj  == 'void':#若不满足，则适应度为无穷
                    fitness  = float('inf')
                    break
                elif  lj+X/tan_beta_j_1-X/tan_beta_j<0:#mj<0
                    fitness  = float('inf')
                    break
                else:
                    Dj-= lj*math.tan(math.radians(1.5))
                    sum_lj+=lj
```

```
                if sum_lj >= Y:
                    fitness = 0
                    for j in range(0,i):
                        fitness += X/math.sin(angle_list[j+1])
                    break
    return fitness
#轮盘赌法筛选个体
import random
def roulette_wheel_selection(population, fitness_values, num_parents):
    # 计算适应度总和
    total_fitness = sum(fitness_values)
    # 计算每个个体的选择概率
    selection_probabilities = [fitness / total_fitness for fitness in fitness_values]

    selected_parents = []
    for _ in range(0,int(num_parents)):
        # 使用轮盘赌法随机选择一个个体
        selected_index = roulette_wheel_spin(selection_probabilities)
        selected_parents.append(population[selected_index])

    return selected_parents

def roulette_wheel_spin(selection_probabilities):
    # 在 0, 1 范围内生成一个随机数
    spin = random.random()
    cumulative_probability = 0.0
    # 根据选择概率进行选择
    for i, probability in enumerate(selection_probabilities):
        cumulative_probability += probability
        if spin < cumulative_probability:
            return i
#交叉和变异
def crossover(parents):
    target_count = count - len(parents)
    children = []
    while len(children) < target_count:
        while True:
            male_index = random.randint(0, len(parents)-1)
            female_index = random.randint(0, len(parents)-1)
            if male_index != female_index:
```

```
                    break
            male = parents[male_index]
            female = parents[female_index]
            left = random.randint(0, len(male) - 2)
            gen_male = male[left:]
            gen_female = female[left:]
            child_a = []
            child_b = []
            child_a = male[0:left]+gen_female
            child_b = female[0:left]+gen_male
            children.append(child_a)
            children.append(child_b)
    return children
def mutation(children):
    #把 children 内部的位点交换
    for i in range(len(children)):
        if random.random() < mutation_rate:
                u = random.randint(0, len(children[i]) - 1)
                if children[i][u] =='1':
                    temp_a = children[i]
                    children[i] = temp_a[0:u]+'0'+temp_a[u+1:]
                    break
                else:
                    temp_a = children[i]
                    children[i] = temp_a[0:u]+'1'+temp_a[u+1:]
    return(children)
#get result
def get_result(population):
    grad = [[individual, 1/fitness(individual)] for individual in population]
    grad = sorted(grad, key=lambda x: x[1])
    return grad[0][0], grad[0][1]
#生成初始种群
import random
population = []
count = 20#种群的个体数量
def generate_binary_numbers(num_numbers, num_bits=10):
    binary_numbers = []
    for _ in range(num_numbers):
        random_int = random.randint(0, (1 << num_bits) - 1)
        random_int = random.randint(500, 511)
```

```
            # 将随机整数转换为十位的二进制字符串，并补齐前导零
            binary_str = format(random_int, f'0{num_bits}b')
            binary_numbers.append(binary_str)
    return binary_numbers
for _ in range(count):
    random_binary_numbers = generate_binary_numbers(40)
    individual = ''.join(random_binary_numbers)
    population.append(individual)
fitness_values = []
for individual in population:
    fitness_values.append(1/fitness(individual))
if sum(fitness_values) == 0:
            i = random.randint(0,len(population))
            p = random.randint(0,40)
            population[i]= '0111111111'*40

#主体部分
# 种群内个体数
count = 20
# 进化次数
iter_time = 100
#变异
mutation_rate = 0.1
#初始化种群
i = 0
num_parents = 0.5*count#后续再定
fitness_best_list = []
while i < iter_time:
    # 个体评估与检验
    fitness_values = []
    individual_list = []
    for individual in population:
        fitness_values.append(1/fitness(individual))
    if sum(fitness_values) == 0:
        i = random.randint(0,len(population))
        population[i]= '0111111111'*40
        fitness_values[i] = 1/fitness('0111111111'*40)
    #轮盘赌法选择父本
    parents = roulette_wheel_selection(population, fitness_values, num_parents)
    # 繁殖
```

```
223         children = crossover(parents)
224         # 变异
225         mutation(children)
226         # 更新
227         population = parents + children
228         individual_best, fitness_best = get_result(population)
229         individual_list.append(individual_best)
230         fitness_best_list.append(fitness_best)
231         i = i + 1
232         print('代数:{} 适应度: {}'.format(i,fitness_best))
233 print(individual_best)
234 fit = 1/fitness_best
235 print('最终适应度:{}'.format(fit))
236 #绘图
237 import numpy as np
238 import matplotlib.pyplot as plt
239 original_list = fitness_best_list
240 # 对列表中的每个元素取倒数
241 reciprocal_list = []
242 for i in original_list:
243     if i == 0:
244         reciprocal_list.append(float('inf'))
245     else:
246         reciprocal_list.append(1/i)
247 positions = np.arange(len(reciprocal_list))
248 # 可视化数据
249 plt.figure(figsize=(8, 6))
250 plt.plot(positions, reciprocal_list, marker='o', linestyle='-')
251 plt.xlabel('evolution time')
252 plt.ylabel('fitness')
253 plt.title('fitness evolution map')
254 plt.grid(True)
255 plt.ylim(114800, 114900)
256 plt.show()
257
258 #计算条数
259 import math
260 p = twoBit_to_angle(individual_best)
261 sum = 0
262 X = 2*1852
```

```
m = 0
for i in p:
    sum+= X/math.sin(i)
    if sum > fit:
        break
    m +=1
print('侧线条数为：{}'.format(m))
```

B.3 问题 4 代码

B.3.1 待测海域深度可视化

```
clc,clear,close;
Z = readmatrix('附件.xlsx','Range','C3:GU253');
x = readmatrix('附件.xlsx','Range','C2:GU2');x = 1852*x;
y = readmatrix('附件.xlsx','Range','B3:B253');y = 1852*y;
[X,Y] = meshgrid(x,y);

figure();
subplot(1,2,1);
scatter3(X,Y,Z,1,' filled ');
xlabel('X (东西方向)');ylabel('Y (南北方向)');
zlabel('深度 (m)');
set(gca,'FontSize',18);axis square;
subplot(1,2,2);
pcolor(X,Y,Z);
shading interp; colorbar;
xlabel('X (东西方向)');ylabel('Y (南北方向)');
set(gca,'FontSize',18);axis square;
```

B.3.2 求解代码

```
import pandas as pd
import math
import random

# 读取 Excel 文件
df = pd.read_excel('depth.xlsx')

```

```
# 海域尺寸
north_south_length = 5*1852 # 南北方向长度（海里）
east_west_width = 4*1852 # 东西方向宽度（海里）

theta = 120
# 矩形块参数
rectangle_width = 1852*0.1 # 矩形块宽度（海里）
rectangle_height = 1852*0.2 # 矩形块高度（海里）
overlap_rate_min = 0.15 # 重叠率
overlap_rate_max = 0.25

# 遍历矩形块
for overlap_rate in range(int(overlap_rate_min * 100000), int(overlap_rate_max * 1000000)
    + 1):
    overlap_rate = overlap_rate / 100000
    # 测线长度和重叠率列表
    line_lengths = []
    overlap_rates = []
    for i in range(1, int(north_south_length / rectangle_height) + 1):
        for j in range(1, int(east_west_width / rectangle_width) + 1):
            # 获取当前矩形块的深度值
            depth = df.iloc[i - 1, j - 1]
            alpha = math.tanh(depth / (0.03 * i))
            k = (math.cos(theta / 2 + alpha) * math.cos(theta / 2 - alpha)) / (
                    math.sin(theta) * math.cos(alpha) ** 2)  # 常数
            # 计算测线长度
            if i == 1 and j == 1:
                line_length = 0.1
            else:
                line_length = rectangle_height*(overlap_rate * df.iloc[i - 2, j - 1]) / (
                        overlap_rate * math.tan(math.radians(alpha)) + k)
            line_lengths.append(line_length)

    # 计算指标
    total_length = sum(line_lengths)
    missed_area_percentage = (1 - len(line_lengths) / (
            (north_south_length / rectangle_height) * (east_west_width / rectangle_width
                ))) * 100
    overlap_length = sum([length for length in line_lengths if length > rectangle_height *
        overlap_rate])
```

```
45
46      # 以仿真洒点去计算覆盖面积
47      point_scattering = 100000 # 模拟洒点数量
48      covered_area = 0
49
50      for _ in range(point_scattering):
51          x = random.uniform(0, east_west_width)
52          y = random.uniform(0, north_south_length)
53
54          for i in range(1, int(north_south_length / rectangle_height) + 1):
55              for j in range(1, int(east_west_width / rectangle_width) + 1):
56                  if x >= (j - 1) * rectangle_width and x < j * rectangle_width and y >= (
57                          i - 1) * rectangle_height and y < i * rectangle_height:
58                      depth = df.iloc[i - 1, j - 1]
59                      alpha = math.tanh(depth / (0.03 * i))
60                      k = (math.cos(theta / 2 + alpha) * math.cos(theta / 2 - alpha)) / (
61                              math.sin(theta) * math.cos(alpha) ** 2)
62                      line_length = (overlap_rate * depth) / (overlap_rate * math.tan(math.
                            radians(alpha)) + k)
63                      if line_length >= rectangle_height * overlap_rate:
64                          covered_area += rectangle_width * rectangle_height
65
66      coverage_percentage = (covered_area / (north_south_length * east_west_width))
67
68      # 输出结果
69      print("覆盖面积的百分比：", coverage_percentage)
70      print("测线的总长度：", total_length)
71      print("漏测海区占总待测海域面积的百分比：", 100 - coverage_percentage)
72      print("在重叠区域中，重叠率超过 20% 部分的总长度：", overlap_length)
```