Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра дифференциальных и стохастических уравнений

РАБОТА ПРОВЕРЕНА
Рецензент, кандидат технических наук, доцент
_____ / Б.М. Кувшинов /
«___» _____2016 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой дифференциальных и стохастических уравнений, ЮУрГУ, доктор физ.-мат. наук, доцент
_____ / С.А. Загребина /
«___» _____2016 г.

ОПТИМИЗАЦИЯ ЭФФЕКТИВНОСТИ СИСТЕМЫ БЕЗОПАСНОСТИ ОХРАНЯЕМОГО ОБЪЕКТА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2016.129-128.ВКР

Руководитель, канд. физ.-мат. наук., доцент
_____ / С.М. Елсаков /
«___» _____2016 г.

Автор, студент группы ММиКН-293
_____ / В.М. Корсаков /
«30» мая 2016 г.

Нормоконтролер, канд. физ.-мат. наук, доцент
_____ / М.А. Сагадеева /
«___» _____2016 г.

Челябинск 2016

Зинистерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра уравнений математической физики

З А Д А Н И Е

студенту группы ММиКН-293
Корсакову Владиславу Михайловичу
на выпускную квалификационную работу
по направлению 09.04.04 – ПРОГРАММНАЯ ИНЖЕНЕРИЯ

1. **Тема диссертации:** «Оптимизация эффективности системы безопасности охраняемого объекта».
   (Утверждена приказом по университету от «15» апреля 2016г. № 661)
2. **Перечень подлежащих исследованию вопросов**
   2.1. Литературный обзор, история вопроса и результаты предыдущих исследований
   2.2. Постановка задачи
   2.3. Методы исследования и ожидаемые результаты
3. **Календарный план подготовки выпускной квалификационной работы**

Наименование этапов дипломной работы	Срок выполнения этапов работы	Отметка о выполнении
1. Обзор литературы	01.02.16 – 14.02.16	
2. Выбор и разработка методов исследования	15.02.16 – 28.02.15	
3. Получение результатов, формулировка выводов, структурирование текста.	29.02.16 – 15.04.16	
4. Подготовка текста выпускной квалификационной работы	15.04.16 – 15.05.16	
5. Проверка и рецензирование работы руководителем, исправление замечаний	15.05.16 – 25.05.16	
6. Подготовка доклада и текста выступления	26.05.16 – 10.06.16	
7. Внешнее рецензирование	20.05.16 – 05.06.16	
8. Защита дипломной работы	10.06.16 – 20.06.16	

4. **Дата выдачи задания** «01» февраля 2016 г.

Руководитель работы
кандидат ф.- м. наук _____ (подпись) ***Елсаков С.М.***

Задание принял к исполнению _____ (подпись) ***Корсаков В.М.***

УДК – 51-74

Корсаков В.М.

Оптимизация эффективности системы безопасности охраняемого объекта. / В.М. Корсаков. – Челябинск, 2016. – 50 с.

Данная работа посвящена разработке программного комплекса моделирования и оценки эффективности системы безопасности охраняемого объекта, а также алгоритма оптимизации эффективности системы безопасности с заданными параметрами. Программный комплекс разработан с использованием языков C++ и Python для операционной системы Windows.

Библиографический список – 19 наим., иллюстраций – 13, приложений – 1 на 19 листах.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Моделирование и оценка системы безопасности охраняемого объекта заключается в построении математической модели «объект – нарушитель» и оценки вероятности проникновения нарушителя на охраняемый объект. Как правило, моделирование заключается в рассмотрении множества аспектов, таких как физическая защита – для предотвращения непосредственного проникновения на объект; информационная защита – для предотвращения утечки или компрометации критически важных данных; организация персонала – для обеспечения контроля за работниками, и т.д.

Система физической защиты (СФЗ), в свою очередь, представляет собой сложную организацию элементов обнаружения, задержки и реагирования. Задачей нарушителя является достижение цели с минимальной вероятностью быть остановленным СФЗ. В частности, нарушитель может попытаться минимизировать вероятность обнаружения (скрытное проникновение), или компенсировать реакцию службы безопасности (вооруженное нападение). При рассмотрении различных охраняемых объектов следует учитывать специфику предполагаемого нарушителя и его возможные стратегии.

В работе А.В. Вергейчика и В.П. Кушнира [4] рассматривается модель системы физической защиты, основанной на графах, в которой для оценки вероятности проникновения используется алгоритм Дейкстры. В исследовании Я.Н. Немова [8] рассматривается применение графовой модели системы безопасности для объектов ФСИН. Ф.А. Дюран в своей диссертации [11] рассматривает различные подходы к моделированию систем безопасности ядерных объектов. В ряде работ В.В. Башурова [2][3] предлагается использование функции обнаружения нарушителя на территории для моделирования системы безопасности и оценки вероятности проникновения, также в работе Т.И. Филимоненковой и В.В. Башурова [9] предлагается метод оптимизации системы безопасности, основанный на решении вариационной задачи.

В работе предлагается использование однородного алгоритма многоэкстремальной оптимизации применительно к предложенной в [2][3] модели системы безопасности, а также приводится программная реализация данного алгоритма и модели системы безопасности.

1. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ СИСТЕМЫ БЕЗОПАСНОСТИ

1.1. Обзор моделей системы безопасности

В основном системы безопасности рассматриваются как набор критических компонентов, уязвимых для проникновения, например, рубежей обороны или участков повышенной безопасности, между которыми возможны переходы нарушителя, и вероятностей обнаружения на каждом таком переходе. Модель такой системы можно представить в виде графа, вершины которого соответствуют компонентам системы безопасности, а веса ребер – вероятностям обнаружения. Так, в [4] рассматривается модель объекта, охраняемая область которого разбивается на участки однородности, в которых характеристики системы защиты считаются одинаковыми.

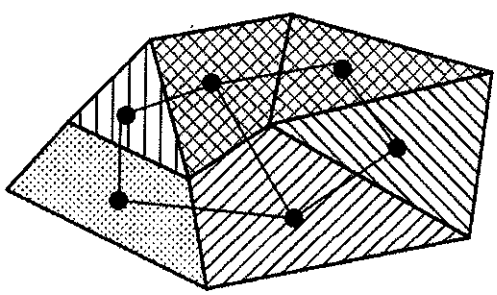

Рис 1. Участки однородности

По модифицированному алгоритму Дейкстры находятся все пути из множества точек начала движения $N = (n_1, n_2, n_i, \dots, n_k)$ (областей однородности, расположенных на периметре объекта) к множеству целевых точек $C = (c_1, c_2, c_i, \dots, c_m)$ (области однородности, в которых находятся критические объекты). Рассматриваемая модель также учитывает возможную реакцию службы безопасности. Итоговая оценка системы вычисляется исходя из времени, необходимого на остановку нарушителя и вычисленных на предыдущем этапе вероятностей проникновения.

В работе [9] рассматривается модель системы безопасности объектов ФСИН и уделяется особое внимание специфики различных объектов. Так, определяются различные типы охраняемых объектов, например: исправительные учреждения, транспорт, сети связи и электроснабжения; а также способы нарушения режима безопасности: открытое столкновение, скрытое проникновение, побег, засады на транспорт и т.д. Для различных охраняемых объектов будут возможны лишь часть из всех рассматриваемых стратегий нарушителя, а также вероятность их применения. Стратегии, в свою очередь, характеризуются возможностью применения различных действий, таких подкоп, подделка документов, взлом и т.д. Каждое из действий также характеризуется вероятностью

применения. Исходя из этих данных, становится возможно вычислить максимальную вероятность проникновения на объект и используемые для этого средства.

Недостатками перечисленных моделей являются заведомое уменьшение числа возможных стратегий поведения нарушителя и особенностей охраняемого объекта. В работе [2] предлагается модель, которая лишена данных недостатков и направлена исключительно на рассмотрение вероятности проникновения нарушителя на объект и не рассматривает сценарий остановки нарушителя, а также предлагается метод оценки системы безопасности.

1.2. Вариационная модель системы безопасности

Рассмотрим следующую модель системы безопасности охраняемого объекта [2][3]: имеется охраняемая область D с границей S_D, в которой расположен охраняемый объект M_0, и средства обнаружения нарушителя (например, наблюдательные вышки, часовые, видеокамеры и т.д.), перемещающегося по области D с какой-либо точки M_D границы S_D и стремящегося достичь объекта M_0. Объекты системы безопасности характеризуются локальными функциями обнаружения $p_i(M, t)$, композиция которых определяет функцию обнаружения $p(M, t)$ на всей охраняемой области.

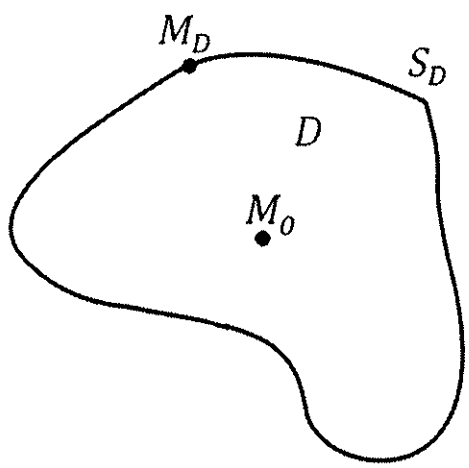

Рис. 2. Модель охраняемой области

Функция обнаружения $p(M, t)$

$$p(M, \tau)\colon \mathrm{D} \times [0, \infty) \to \mathbb{R} \tag{1}$$

определяется составом и расположением средств обнаружения и обладает следующими свойствами:

1. $p(M(x,y), \tau) \geq 0, \ \forall M(x, y) \in D;$
2. $\int_0^\infty p(M(x,y), \tau) d\tau \leq 1, \ \forall M(x, y) \in D;$
3. $p(M(x,y), \tau)$ непрерывна при $\tau \to 0$.

Вероятность обнаружения неподвижного нарушителя в точке M за время t:

$$P(M,t) = \int_0^t p(M,\tau)d\tau \tag{2}$$

Предполагая, что нарушитель двигается по траектории Г, разбитой на участки $\Delta\Gamma_i$, определим вероятность необнаружения нарушителя на i-м участке траектории следующим образом:

$$\Delta p_i = 1 - \int_0^t p\big(M(x,y),\ \tau\big)d\tau, \tag{3}$$

где t – время пребывания нарушителя на участке $\Delta\Gamma_i$.

Тогда вероятность необнаружения нарушителя на всей траектории движения вычисляется как произведение вероятностей необнаружения на каждом участке:

$$P(\Gamma) = \prod_{i=1}^{k} \Delta p_i, \tag{4}$$

где k – количество участков в разбиении.

Логарифмируем обе части (4):

$$\ln\big(P(\Gamma)\big) = \sum_{i=1}^{k} \ln(\Delta p_i), \tag{5}$$

Принимая время нахождения t_i на i-м участке разбиения бесконечно малым, и используя равенство (3) заменим правую часть (5):

$$\ln\left(1 - \int_0^{t_i} p(M(x,y),\tau)d\tau\right) \approx -\int_0^{t_i} p(M(x,y),\tau)d\tau, \tag{6}$$

$$\int_0^{t_i} p(M(x,y),\tau)d\tau \approx p(M(x,y),0)\Delta\tau_i, \tag{7}$$

где $\Delta\tau_k = t_k = ds_k/V_i(x,y)$,

t_i – время пребывания нарушителя на участке $\Delta\Gamma_i$,

$V_i(x,y)$ – скорость перемещения нарушителя на участке.

Таким образом получаем следующую оценку логарифма вероятности обнаружения нарушителя на траектории Г:

$$\ln\big(P(\Gamma)\big) = -\sum_{i=1}^{k} p(M(x,y),0)\Delta\tau_i,$$

$$\ln\big(P(\Gamma)\big) = -\sum_{i=1}^{k} \frac{p(M(x,y),0)}{V_i(x,y)} ds_k,$$

$$\ln\big(P(\Gamma)\big) = -\int_{\Gamma} \frac{p(M(x,y),0)}{V_i(x,y)} ds. \tag{9}$$

Определим функцию обнаружения движущегося нарушителя, считая, что его движение не останавливается:

$$p_0(x,y) = p\big(M(\mathrm{x},\mathrm{y}),\ 0\big).$$

Обозначим

$$f(x,y) = \frac{p\big(M(\mathrm{x},\mathrm{y}),\ 0\big)}{V(x,y)}.$$

Таким образом, получаем

$$\ln\big(P(\Gamma)\big) = -\int_{\Gamma} f(x,y)\, ds,$$

$$P(\Gamma) = \exp(-\int_{\Gamma} f(x,\ y)\, ds). \tag{10}$$

Функционал (10) определяет вероятность обнаружения нарушителя при его движении по траектории Г. Эффективность системы безопасности можно оценить максимумом данного функционала, а траектория $\bar{\Gamma}$, на которой достигается максимум, будет наиболее уязвимым путем проникновения. Для нахождения оценки эффективности необходимо решить задачу:

$$P_{\bar{\Gamma}} = \max_{\Gamma}\left(\exp\left(-\int_{\Gamma} f(x,\ y)\, ds\right)\right), \tag{11}$$

или же

$$I_{\bar{\Gamma}} = \min_{\Gamma}\left(\int_{\Gamma} f(x,\ y)\, ds\right). \tag{12}$$

1.3. Волновой метод оценки системы безопасности

Решение задач (11) и (12) аналитическими методами не представляется возможным ввиду того, что функция $f(x,\ y)$ может быть разрывной или же задаваться численно. Для

этого предлагается «волновой» метод, построенный на аналогии распространения света в оптически неоднородной среде [2][9].

Зададим некоторую малую величину Δ, называемую «шагом волны». Обозначим как $V_\Delta(M)$ множество точек области D, до которых существует путь Γ из точки M, причем $p(\Gamma) \leq \Delta$. Пусть $X_i \subset D, L_i -$ граница множества X_i, и примем $X_0 = S_D$. Будем последовательно строить множества X_i по правилу

$$X_i = X_{i-1} \bigcup_{M \in L_{i-1}} V_\Delta(M).$$

Множество X_i соотвествует точкам множества D, которые могут быть достигнуты с границы за i шагов, на каждом из которых оценка вероятности обнаружения увеличивается не более чем на Δ.

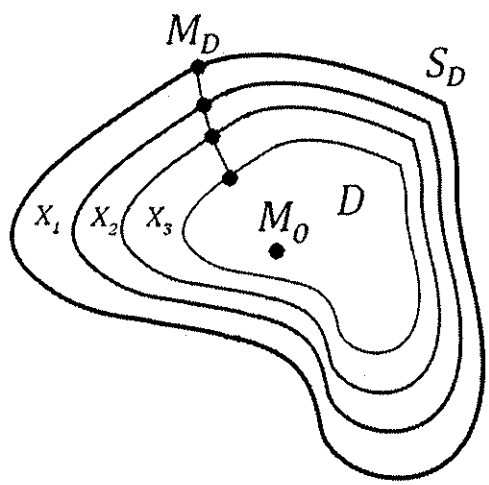

Рис 3. Фронт волны X_i

Для нахождения оценки системы безопасности проводится построение множеств X до тех пор, пока не будет достигнут охраняемый объект. Пусть охраняемый объект достигается на волне с номером k. Тогда можно оценить значение функционала (11):

$$(k-1) \cdot \Delta \leq \mathrm{p}(\Gamma) \leq k \cdot \Delta.$$

Из определения фронта волны следует, что для любой точки $M_i \in L_i$ существует путь $\tilde{\Gamma}_{m_i}$ соединяющий M_i и некоторую точку $M_{i-1} \in L_{i-1}$ для которого $\mathrm{p}(\tilde{\Gamma}_{m_i}) = \Delta$. Таким образом, последовательное построение множеств $X_i \backslash X_{i-1}$при заданном достаточно малом Δ приведет к нахождению набора точек $\bar{\Gamma}$, соединяющих фронты волн и аппроксимирующих искомую траекторию Γ, причем

$$\int_{\bar{\Gamma}} f(x, y)\, ds = (k-1) \cdot \Delta + \delta.$$

Для программной реализации описанного метода оценки были применены дискретизация модели и алгоритм поиска в ширину.

1.4. Дискретизация модели

Для реализации описанной модели системы безопасности и волнового метода оценки необходимо провести её дискретизацию. Для этого область D покрывается прямоугольной сеткой $n \times m$, в узлах которой вычисляется значение функции $p(M(x,y), \tau)$. Полученную сетку со значениями функциями будем рассматривать как граф, веса ребер которого соответствуют вероятности обнаружения нарушителя при переходе его по смежным вершинам.

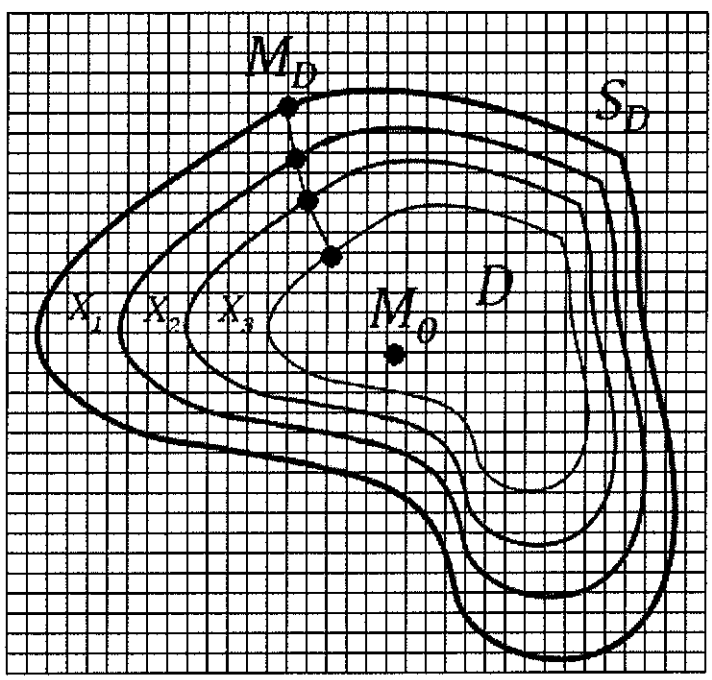

Рис. 4. Покрытие области сеткой

Алгоритм поиска в ширину (BFS) применяется для поиска кратчайших путей и компонент связности в графах.[7] Пусть требуется найти кратчайший путь от множества узлов S до остальных или одного конкретного. Просмотр ведется от узлов из S во все возможных направлениях с добавлением одного «уровня» за раз. Первый уровень поиска формируется путем включения всех соединенных ребром с S. Второй уровень формируется путем включения всех узлов соединенных ребром с любым узлом из первого уровня. Просмотр продолжается до того момента, когда очередная попытка не обнаружит ни одного нового узла.

Уровни L_1, L_2, L_3, ... , создаваемые алгоритмом BFS, более точно определяются следующим образом:

- Уровень L1 состоит из всех узлов, являющихся соседями s.
- Если предположить, что определены уровни $L_1, ..., L_j$, то уровень L_{j+1} состоит из всех узлов, не принадлежащих ни одному из предшествующих уровней и соединенных ребром с узлом уровня L_j.

Заметим, что метод поиска в ширину близок к описанному волновому методу оценки системы безопасности. Построение волн в алгоритме оценки схоже с построением слоев поиска узлов в BFS. Важным отличием является то, что распространение волн в методе оценки идёт в первую очередь по пути наименьшей вероятности обнаружения, в то время как в алгоритме BFS все узлы считаются равноправными.

При инициализации алгоритма в очередь рассматриваемых вершин добавляются все узлы графа, лежащие на границе S_D, упорядоченные по значению функции $p(M, t)$. На каждом этапе алгоритма из очереди извлекается очередной узел и рассматриваются все его смежные узлы, при этом вычисляется итоговая вероятность обнаружения при переходе из данного узла в соседний. В случае, если вычисленная вероятность оказывается ниже, чем вычисленная ранее, то итоговое значение вероятности проникновения в данный узел обновляется, а все его смежные узлы добавляются в очередь рассмотрения в соответствии с их весами.

Алгоритм продолжает свою работу до тех пор, пока не будет достигнут узел графа, соответствующий охраняемому объекту, или не будут перебраны все доступные для посещения узлы. Итоговое значение вероятности, соответствующее узлу с охраняемым объектом (или максимальное из них, в случае, если таких узлов несколько) будет соответствовать максимуму функционала (2), и являться оценкой эффективности системы безопасности.

1.4. Задача оптимизации системы безопасности

Средства обнаружения нарушителя, составляющие систему безопасности, обладают различными параметрами (например, координатами местонахождения, направлением обзора и т.д.), которые будут оказывать существенное влияние на функцию $p(x, y)$[9]. Так, например, наблюдательная вышка будет иметь два параметра – координаты местоположения, наблюдательная камера будет иметь три параметра – координаты местоположения и направление обзора.

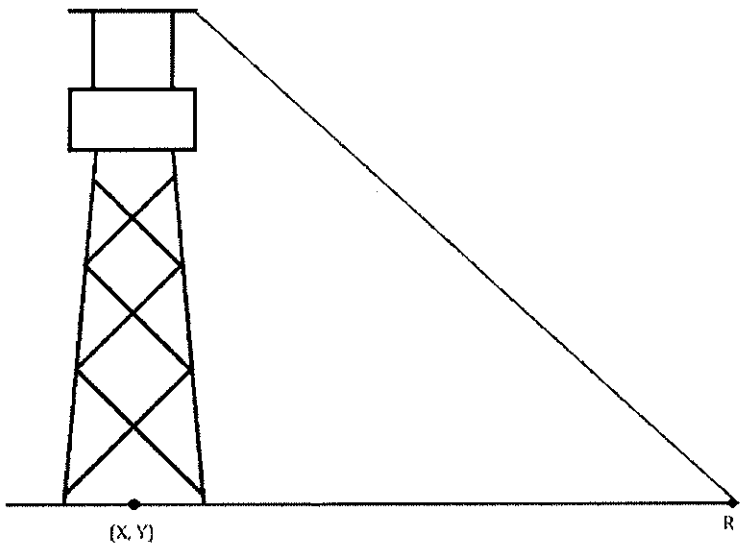

Рис. 4. Параметры наблюдательной вышки

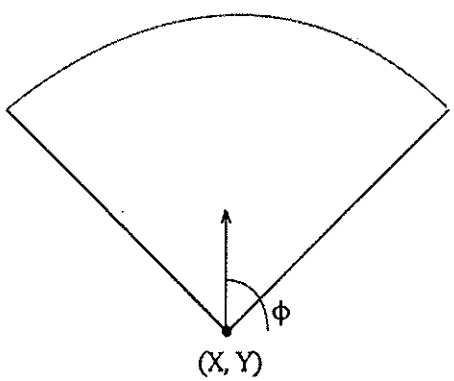

Рис. 5. Параметры наблюдательной камеры

Пусть $\alpha = (\alpha_1, \dots, \alpha_n), \alpha_i \in A_i \subset \mathbb{R}$ – набор параметров всех объектов системы безопасности. Рассмотрим параметризованную функцию обнаружения:

$$p(\Gamma, \alpha) = \int_{\Gamma} p_0(x, y, \alpha) ds \tag{13}$$

Для обеспечения максимальной безопасности охраняемого объекта необходимо построить систему безопасности, при которой вероятность проникновения нарушителя на объект минимальна. При заданном наборе параметров α с помощью описанного выше волнового метода оценки определяется траектория Γ_α, являющаяся наиболее уязвимым местом системы безопасности:

$$F(\alpha) = \min_{\Gamma} p(\Gamma, \alpha), \tag{14}$$

а значение $F(\alpha)$ назовем эффективностью системы безопасности.

Значение

$$F = \max_{\alpha} F(\alpha) = \max_{\alpha} \min_{\Gamma} p(\Gamma, \alpha),$$
$$\alpha = (\alpha_1, \dots, \alpha_n), \alpha_i \in \mathrm{A}_i \subset \mathbb{R}, \tag{15}$$

будет являться оценкой оптимальной системы безопасности при заданном наборе средств обнаружения. Решение же задачи (15) представляет собой задачу однокритериальной оптимизации.

1.5. Программные комплексы для решения задач оптимизации

Для решения задач многоэкстремальной оптимизации существуют программные комплексы, среди большого числа которых можно отметить следующие:

1. BARON [19] — программный комплекс для решения непрерывных, целочисленных и смешанных задач оптимизации. BARON использует метод ветвей и границ, метод ветвей и упрощений (branch and reduce), двойственность и интервальный анализ для решения задач многоэкстремальной оптимизации. Разработан под руководством N. Sahinidis, университет Карнеги—Меллон, США.

2. BNB-Solver [1] — программный комплекс для решения дискретных и непрерывных задач многоэкстремальной оптимизации. Совместно с BNB-Grid позволяет организовывать параллельное решение задач многоэкстремальной оптимизации на разнородных вычислительных системах. Разработан А. П. Афанасьевым, М. А. Посыпкиным (Институт системного анализа РАН) и Ю. Г. Евтушенко, И. Х. Сигалом (Вычислительный центр РАН).

3. IOSO [12] — программный комплекс для решения задач многоэкстремальной, многомерной и многокритериальной оптимизации для целевых функций вида “черный ящик”. Для решения задач используется построение вспомогательной модели целевых функций с использование различных методов: адаптивные алгоритмы регрессионного анализа, модифицированный метод группового учета аргумента, искусственные нейронные сети. Разработан под руководством И. Н. Егорова, ЗАО "Сигма Технология".

4. LGO [17] — программный комплекс для решения задач многоэкстремальной оптимизации с учетом ограничений для целевых функций вида “черный ящик”. Используются алгоритмы: локального поиска, метод ветвей и границ комбинированный с локальным поиском, рандомизированный метод глобального поиска комбинированный с локальным поиском, рандомизированный мультистартный метод глобального поиска

недифференцируемой. Функция $f(x)$ может быть задана алгоритмически, а её вычисление в одной точке – вычислительно трудоемкая задача.

Любой последовательный алгоритм, который решает задачу (16) вычисляет значения целевой функции в точках x_1, x_2, ... допустимого множества X. Обозначим эту последовательность как $\{x_i\}_{i=1}^{k}$, где k — число проведенных испытаний целевой функции.

Будем полагать, что алгоритм относится к классу однородных алгоритмов многоэкстремальной оптимизации, если одновременно выполнены три условия[6]:

1. Алгоритм является однородным, т.е. для любых двух целевых функций, различающихся на константу, последовательности точек испытаний $\{x_i\}_{i=1}^{k}$совпадают.
2. После каждого испытания целевой функции $f(x)$ определены функции $m_k(x)\equiv m(x, x_1, f(x_1), ..., x_k, f(x_k))$ и $s_k(x)\equiv s(x, x_1, f(x_1), ..., x_k, f(x_k))$, которые удовлетворяют следующим условиям:
   1) $m_k(x_i) = f(x_i), i = \overline{1,k}$;
   2) $s_k(x_i) = 0, i = \overline{1,k}$;
   3) $s_k(x) > 0,\ x \neq x_i, i = \overline{1,k}$;
   4) $m_k(x), s_k(x)$ – липшецевы.
3. Алгоритм может быть представлен в виде:

Исходные данные: целевая функция $f(x)$, алгоритм выбора начальных точек $x_i \in X$, $i = \overline{1,M}$ и их количества M, алгоритм построения функций $m_k(x)$ и $s_k(x)$, функция-характеристика $P(m_k(x), s_k(x)): \mathbb{R}^2 \to \mathbb{R}$, алгоритм решения вспомогательной задачи, условие останова ϕ.

Результат: минимальное найденное значение целевой функции $\min\limits_{i=\overline{1,k}} f(x_i)$, $\arg \min\limits_{x_i, i=\overline{1,k}} f(x_i)$.

Шаг 1. Выбрать начальные точки $x_i \in X, i = \overline{1,M}$. Вычислить $f(x_i), i = \overline{1,M}$. Положить $k = M$.

Шаг 2. Построить функции $m_k(x)$ и $s_k(x)$.

Шаг 3. Найти решение задачи

$$x_{k+1} = \arg\min_{x \in X} P\big(m_k(x), s_k(x)\big),$$

где $P\big(m_k(x), s_k(x)\big)$ – функция характеристика,

$m_k(x), s_k(x)$ – функции, которые описывают модель целевой функции.

Шаг 4. Вычислить $f(x_{k+1})$, положить $k = k + 1$.

2. ПРОГРАММНЫЙ КОМПЛЕКС ОПТИМИЗАЦИИ ЭФФЕКТИВНОСТИ СИСТЕМЫ БЕЗОПАСНОСТИ

Описанные модель охраняемого объекта, волновой метод оценки и алгоритм оптимизации были реализованы в виде программного комплекса, состоящего из трёх модулей:

1. ModelAssessment – модуль, строящий модель системы безопасности с заданными параметрами и производящий оценку с помощью волнового метода;
2. ModelVisualizer – модуль, отвечающий за визуализацию модели с заданными параметрами.
3. ModelOptimization – модуль, реализующий алгоритм оптимизации системы безопасности;

2.1. ModelAssesment

Данный модуль представляет собой консольное приложение, разработанное на языке C++, и предназначенное для построения модели системы безопасности и оценки вероятности проникновения нарушителя на объект. Использование языка C++ позволяет добиться повышения производительности за счет использования оптимизирующих компиляторов.

Данные о параметрах охраняемой области считываются из файла field.txt. Параметры модели, разделенные пробелом, указываются в первой строке файла в следующем порядке:

1. width – ширина охраняемой территории;
2. height – высота охраняемой территории;
3. n – число узлов сетки в разбиении по X;
4. m – число узлов сетки в разбиении по Y.

Данные об объектах охранной системы считываются из файла objects.txt. В первой строке файла указывается n – количество объектов охранной системы типа «Вышка». Каждая из следующих n строк содержат в себе параметры вышки, а именно координаты расположения (X, Y), разделенные пробелом. После в отдельной строке указывается число m – количество объектов типа «Камера». В следующих m строках указываются параметры камер, а именно координаты расположения (X, Y) и угол направления обзора R, указанный в радианах.

При запуске программы считываются данные о модели, после чего производится построение сетки значений функции обнаружение и вычисление оценки вероятности проникновения в каждый узел построенной сетки. Полученные значения выводятся в файлы outputModel.txt и modelAssessment.txt соответственно. Оценка вероятности проникновения нарушителя на объект выводится после выполнения программы в консоль.

2.2. ModelVisualizer

Данный модуль представляет собой скрипт для Matlab/Octave, и предназначен для визуализации данных модели, полученных при выполнении модуля ModelAssesment.

Данные о модели считываются из файла outputModel.txt, данные об оценках вероятности проникновения нарушителя считываютя из файла modelAssessment.txt. Результатом выполнения модуля является отображения на экране полученных данных в графическом виде.

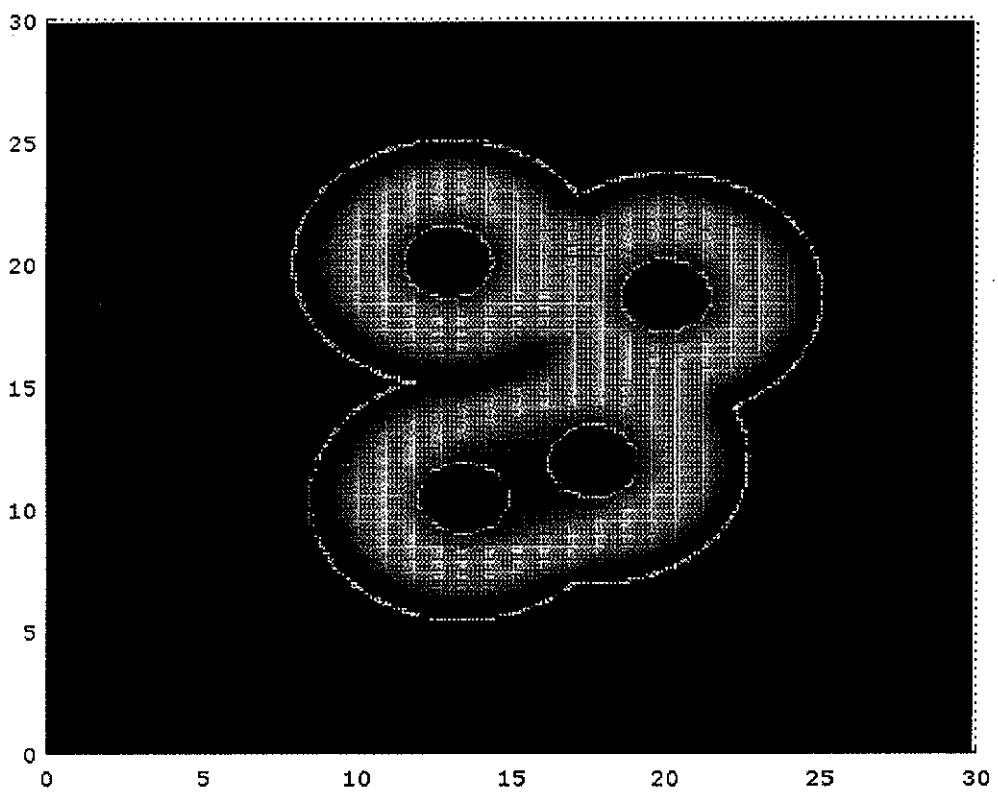

Рис. 5. Визуализация функции обнаружения

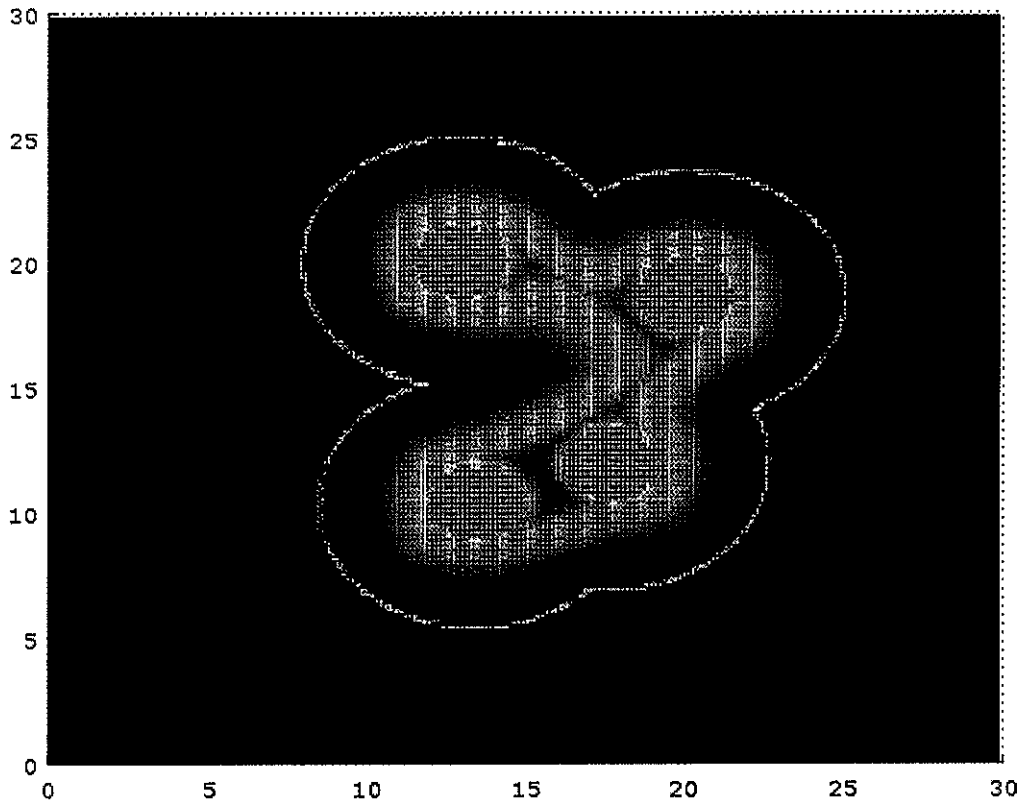

Рис. 6. Визуализация итоговой оценки вероятности обнаружения

2.3. ModelOptimization

Данный модуль представляет собой программу на языке Python, реализующую описанный однородный алгоритм многоэкстремальной оптимизации.

Параметры алгоритма считываются из файла parameters.txt и включают в себя:

1. n – количество первоначальных вычислений значения целевой функции;
2. m – количество уточняющих итераций алгоритма;
3. p – первоначальная плотность размещения точек;
4. r – показатель уменьшения плотности размещения точек.

Входными данными для работы алгоритма являются количество параметров модели и их граничные значения. Данные считываются из файла input.txt. Первая строка файла содержит число n – общее число параметров модели, далее указываются n строк, каждая из которых содержит граничные значения параметра, разделенные пробелом.

Результат работы модуля сохраняется в файл output.csv, представляющий собой табличный документ, каждая строка которого содержит значения параметров модели и вычисленную оценку вероятности проникновения при заданных параметрах.

2.4. Тестирование программы

Программный комплекс проверялся на тестовой модели со следующими параметрами:

Таблица 1. Параметры модели

Параметр	**Значение**
Размер охраняемой области	30x30
Размер сетки дискретизации	200x200
Количество объектов охранной системы	4

Таблица 2. Параметры алгоритма

Параметр	**Значение**
Первоначальная плотность размещения точек	0.75
Показатель уменьшения плотности размещения	1.0
Первоначальное количество вычислений целевой функции	24
Количество уточняющих вычислений	24

В ходе работы алгоритма было проверено в общей сложности 48 вариантов размещения объектов.

Таблица 3. Итерации алгоритма

I вышка		II вышка		III вышка		IV вышка		Вероятность проникновения нарушителя
X	Y	X	Y	X	Y	X	Y	
17.39130	16.52174	15.65217	10.43478	11.30435	15.65217	10.86957	16.95652	0.00043
13.06096	14.38947	15.35010	14.86165	11.45102	17.44214	16.05420	14.14117	0.00081
18.54778	16.18187	11.10570	17.83197	16.08651	12.31020	11.88505	12.64587	0.00087
14.47583	15.56476	16.38820	17.30322	16.16887	11.30800	10.94300	17.20548	0.00088
18.59559	14.36364	13.78644	14.39386	15.73719	13.78372	15.18677	16.69153	0.00089
16.95652	19.56522	14.34783	15.21739	20.00000	10.86957	17.82609	15.21739	0.00093
15.77946	11.47041	17.14565	14.68007	15.94921	14.24344	14.78720	15.45307	0.00101
15.50564	12.10113	17.13837	14.54183	11.40404	17.31147	10.99337	17.53795	0.00110
17.93510	16.90733	13.80012	14.59392	19.76460	10.88340	13.98540	14.90731	0.00111
18.07812	16.02212	11.67850	11.26248	14.12257	13.15650	14.36864	16.19482	0.00117
13.09619	12.19809	14.93508	11.43754	10.41552	16.19052	13.25461	15.19740	0.00118
19.56522	10.00000	16.08696	17.39130	16.08696	12.17391	12.17391	13.91304	0.00123
14.78261	13.91304	16.95652	16.52174	10.00000	16.52174	10.00000	11.30435	0.00128
18.69565	11.73913	13.47826	13.04348	10.86957	16.95652	16.95652	19.13043	0.00133
13.04348	10.43478	20.00000	14.78261	13.47826	18.26087	14.34783	18.69565	0.00202
15.61077	16.20085	11.52992	18.71447	12.51194	13.78868	11.56417	11.64897	0.00271
13.47826	14.34783	18.69565	10.00000	19.13043	14.34783	19.13043	14.34783	0.00734

18.26087	15.21739	11.30435	11.30435	18.26087	15.21739	11.30435	17.82609	0.01087
16.08696	14.78261	10.43478	12.17391	13.04348	13.47826	18.26087	10.43478	0.01698
18.25203	14.49804	10.34611	11.51968	10.80050	17.90822	14.84853	10.44511	0.03554
16.53620	14.35689	12.33797	10.73885	16.05668	16.86933	18.17077	13.46337	0.03866
17.43371	15.91823	10.21417	10.92242	19.38548	13.48368	15.37144	18.78385	0.04024
14.30294	13.02746	18.67184	10.84706	10.68408	17.70193	12.95375	14.02895	0.04723
18.71901	14.68314	10.48335	19.83268	15.79883	12.27262	10.23477	19.16713	0.05916
18.52600	18.74053	10.58040	18.86620	15.82405	10.17986	14.58389	10.99299	0.05937
16.16397	12.80474	11.33409	11.51736	10.30652	17.30664	19.50236	13.68716	0.06336
10.00000	19.13043	15.21739	19.13043	13.91304	12.60870	12.60870	16.08696	0.06706
11.30435	17.82609	12.17391	11.73913	12.17391	19.56522	15.21739	14.78261	0.08358
13.60173	18.44943	16.54293	10.86628	17.90353	12.06312	16.92552	17.03086	0.11713
13.91304	13.47826	17.39130	13.47826	15.65217	10.00000	13.47826	20.00000	0.13409
19.63047	12.96285	13.97103	19.91564	16.57173	10.84947	10.73585	16.54834	0.14709
19.13043	18.69565	13.91304	18.69565	15.21739	16.08696	17.39130	19.56522	0.32226
16.52174	12.60870	10.86957	20.00000	14.78261	19.13043	14.78261	12.60870	0.35878
11.86720	15.01874	16.41894	16.79853	11.89521	14.35366	10.91170	19.19367	0.48280
17.56351	11.88407	13.02811	13.58293	11.19779	17.20114	10.88730	13.06090	0.50558
11.73913	13.04348	12.60870	19.56522	12.60870	13.91304	19.56522	16.52174	0.62536
12.60870	15.65217	11.73913	18.26087	19.56522	13.04348	11.73913	11.73913	0.63742
15.65217	20.00000	13.04348	13.91304	10.43478	10.43478	13.91304	15.65217	0.64044
12.17391	10.86957	14.78261	10.86957	14.34783	11.30435	13.04348	10.86957	1.00000
15.21739	11.30435	19.13043	17.82609	17.82609	14.78261	18.69565	12.17391	1.00000
10.86957	18.26087	16.52174	15.65217	16.52174	18.69565	20.00000	13.47826	1.00000
10.43478	12.17391	10.00000	14.34783	17.39130	17.39130	15.65217	18.26087	1.00000
20.00000	16.08696	17.82609	12.60870	16.95652	17.82609	16.08696	10.00000	1.00000
14.34783	16.95652	18.26087	16.08696	18.69565	20.00000	10.43478	17.39130	1.00000
17.82609	17.39130	19.56522	16.95652	11.73913	11.73913	16.52174	13.04348	1.00000
17.79301	10.16138	19.73708	10.42068	19.03024	12.11361	15.37943	19.65193	1.00000
16.96203	18.16506	10.34934	12.77138	12.86244	19.82349	11.81135	18.78730	1.00000
12.04680	19.38832	12.07195	10.00904	12.69936	16.89574	16.13106	11.27597	1.00000

На рисунках 7-10 приведена визуализация полученной модели системы безопасности на двух начальных итерациях. На рисунках 11 и 12 приведена найденная модель системы безопасности с наименьшей вероятностью проникновения нарушителя на объект.

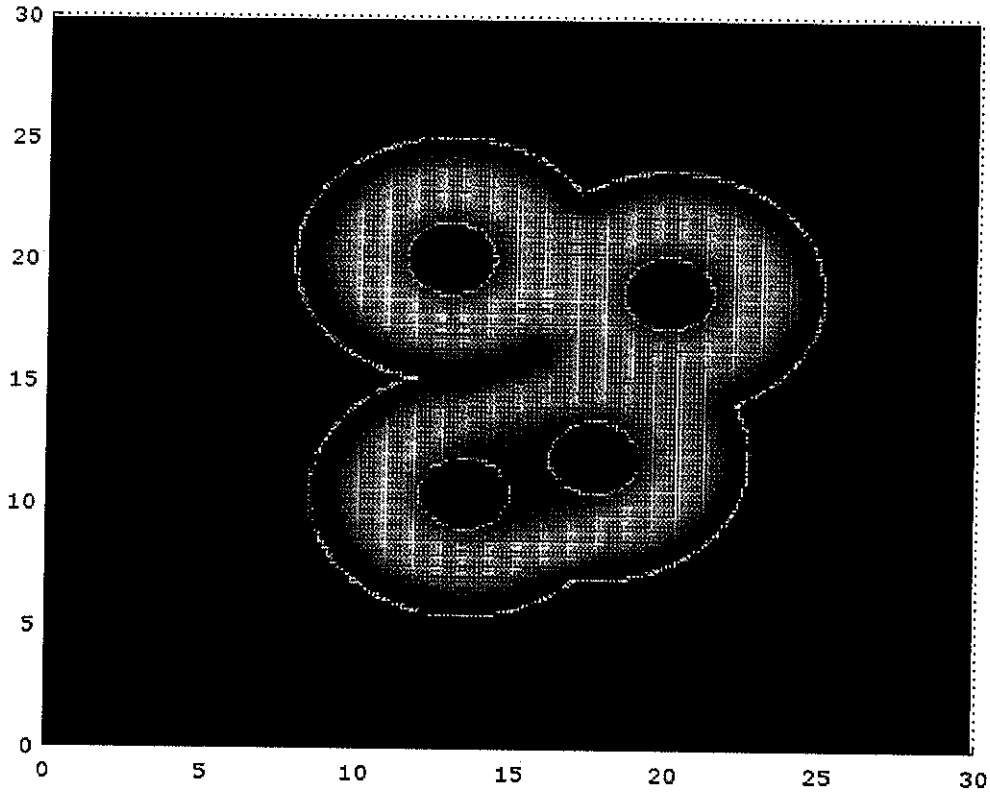

Рис. 8. Функция обнаружения на начальной итерации

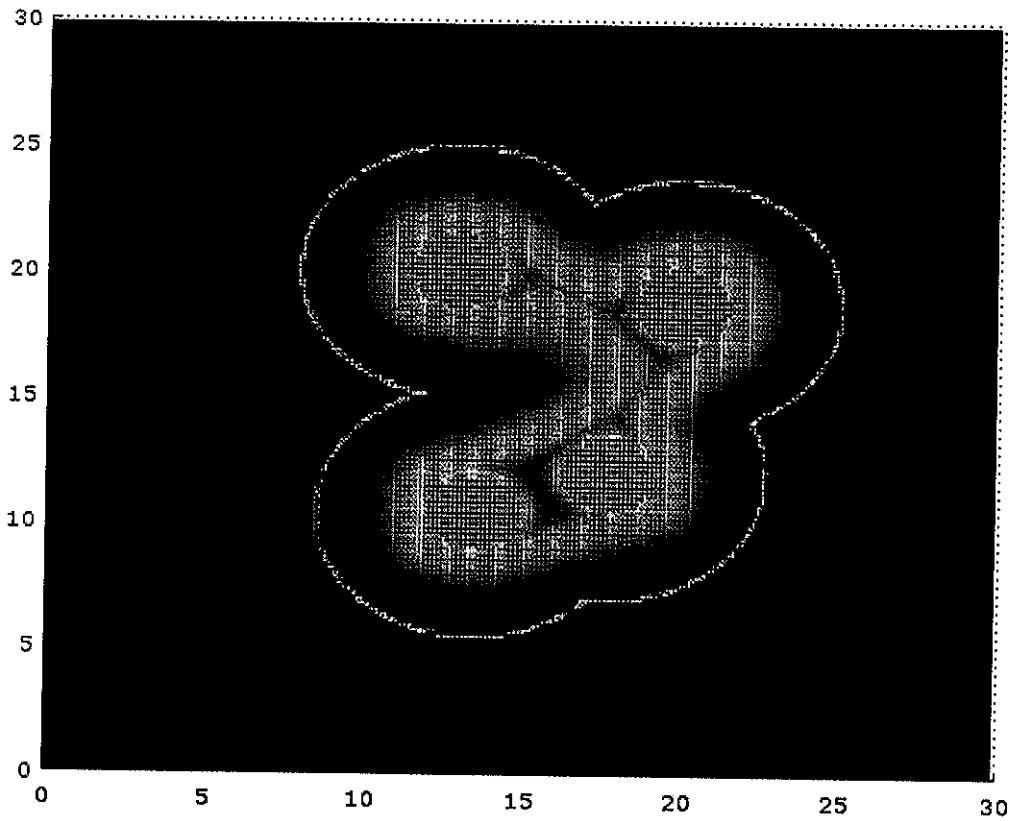

Рис. 9. Оценка вероятности обнаружения на начальной итерации

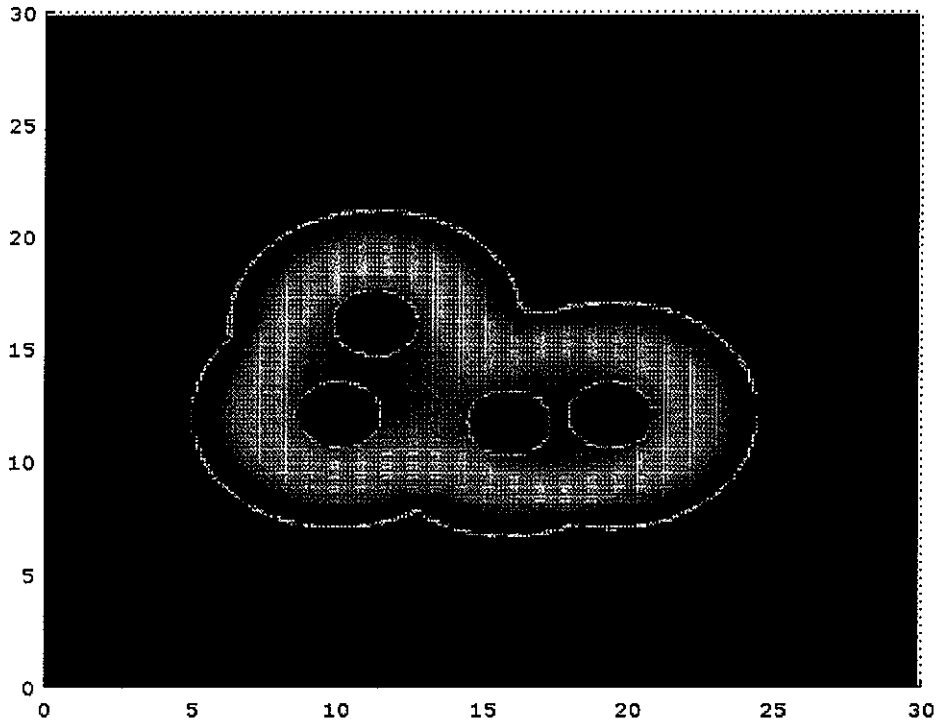

Рис. 10. Функция обнаружения на начальной итерации

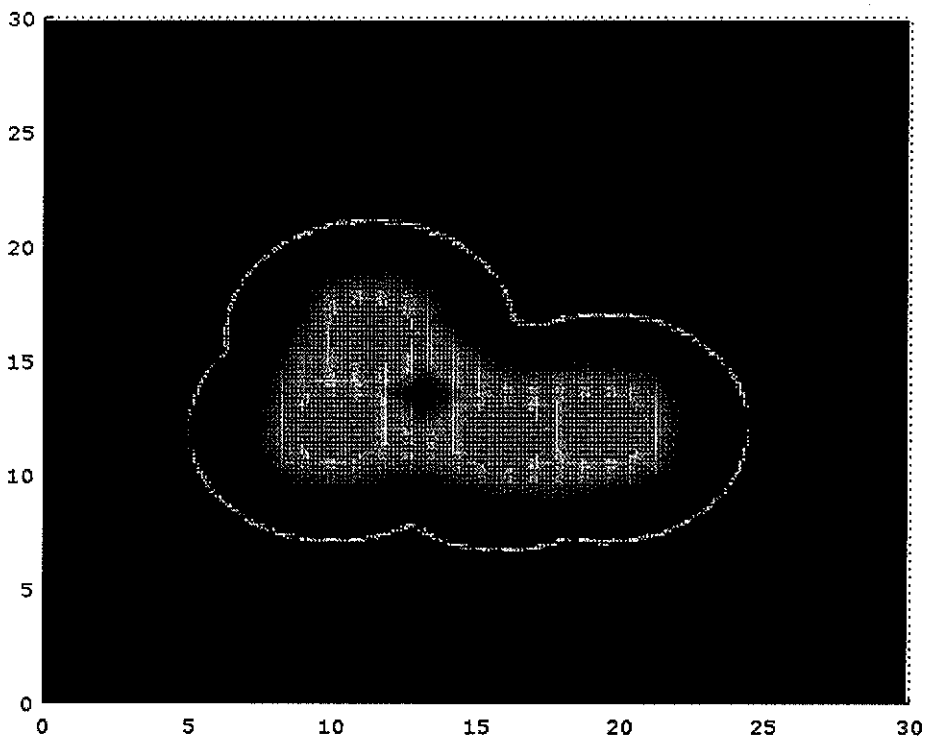

Рис. 11. Оценка вероятности обнаружения на начальной итерации

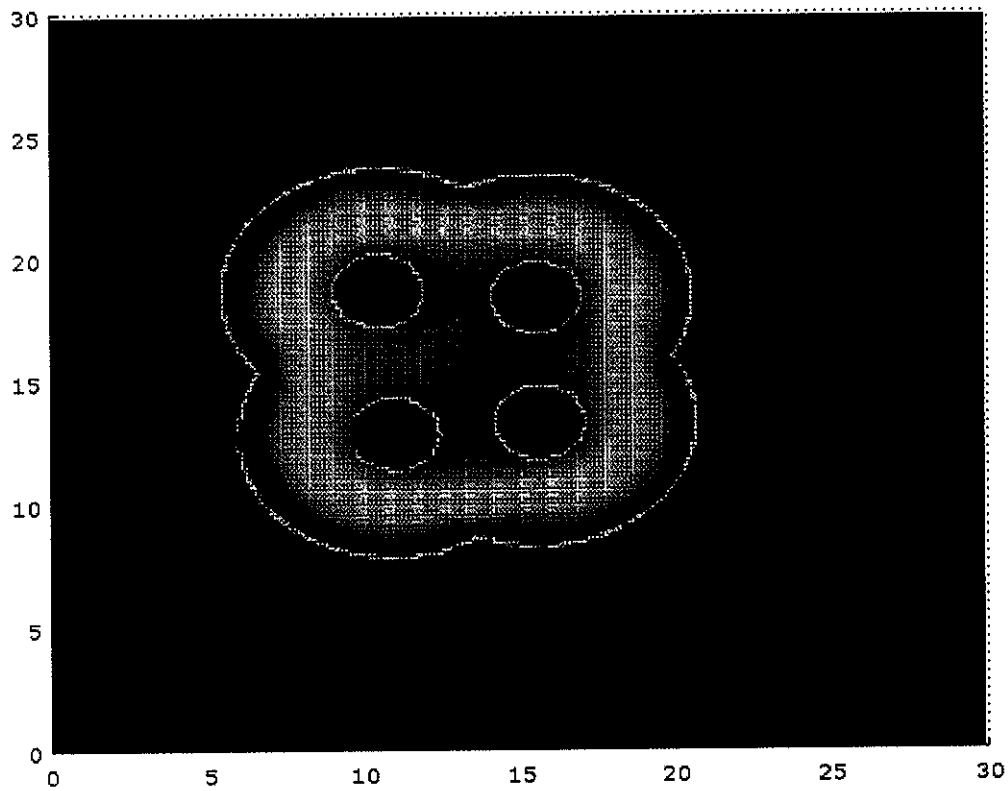

Рис. 12. Функция обнаружения при оптимальных значениях параметров

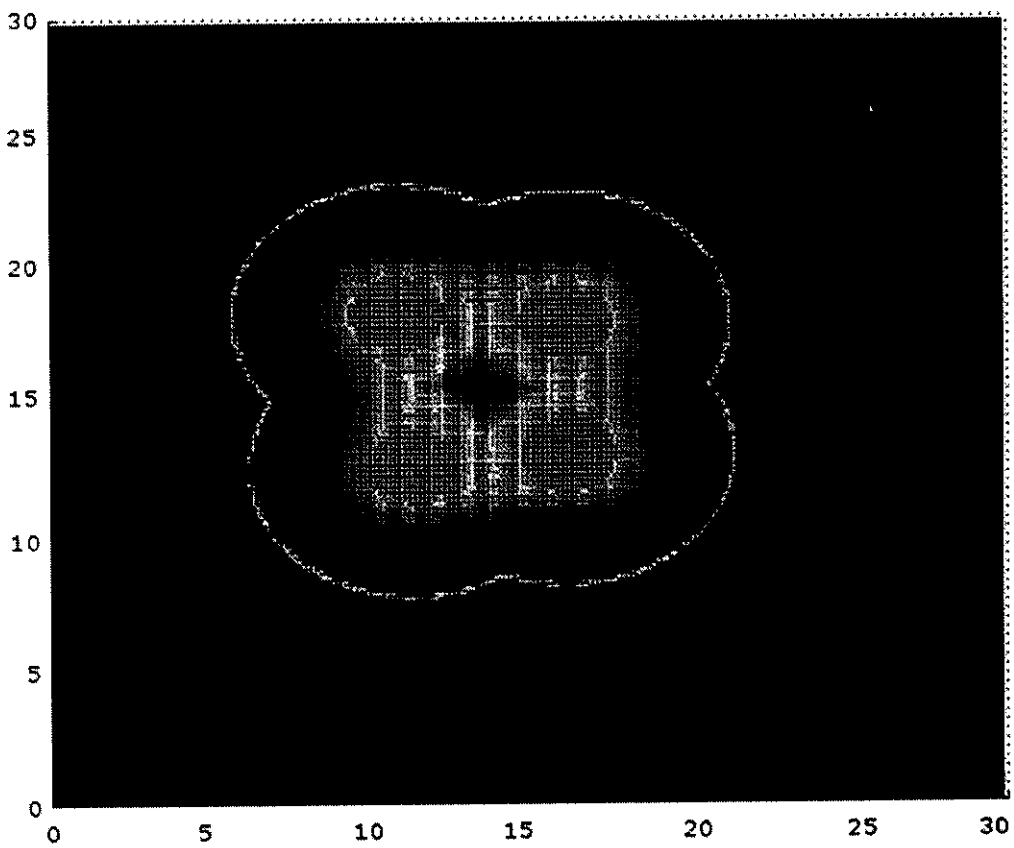

Рис. 13. Оценка вероятности обнаружения при оптимальных значениях параметров

ЗАКЛЮЧЕНИЕ

Были рассмотрены существующие модели и подходы к оценке эффективности систем безопасности. Выявлено, что модель, основанная на функции обнаружения, предложенная В.В. Башуровым в [2], обладает рядом преимуществ перед существующими моделями, основанными на графах.

Рассмотрены существующие программные комплексы для решения задач многоэкстремальной оптимизации и предложено применение однородного алгоритма многомерной оптимизации к задаче оптимизации эффективности системы безопасности.

Разработан программный комплекс, реализующий построение модели системы безопасности с заданными параметрами объектов, а также производящий оптимизацию эффективности системы безопасности, с возможным распараллеливанием вычислений для повышения быстродействия.

Возможна дальнейшая модификация как модели системы безопасности – расширение списка типов охранных объектов, учёт поведения нарушителя со временем, так и алгоритма оптимизации – использование оптимального начального покрытия точек, подбор иной функции интерполяции и т.д.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Афанасьев А.П., Евтушенко Ю.Г., Посыпкин М.А., Сигал И.Х. Решение задач глобальной оптимизации большой размерности на многопроцессорных комплексах и грид-системах // Тр. конф. «Научный сервис в сети Интернет'2008: решение больших задач» / М.: Изд-во МГУ, 2008. – С. 122-130.
2. Башуров, В. В. Математические модели безопасности / В.В. Башуров, Т.И. Филимоненкова. – Новосибирск: Наука, 2009. – 86 с.
3. Башуров, В.В. Применение методов геометрической оптики к решению задачи безопасности объекта / В.В. Башуров // Вычисл. технологии. – Новосибирск, 2006. – Т. 11, №4. – С. 23-28.
4. Вергейчик А. В., Кушнир В. П. Моделирование систем физической защиты // Доклады ТУСУР. – 2008. – №2 (18). Ч. 1. – С. 7-8.
5. Гарсиа М. Проектирование и оценка систем физической защиты. / М. Гарсиа. – М.: Мир: ООО «Издательство АСТ», 2002. – 386 с.
6. Елсаков С.М. Однородные алгоритмы многоэкстремальной оптимизации и модели липшицевых целевых функций: дис. ... канд. физ.-мат. наук / С.М. Елсаков. – Челябинск: Южно-Уральский государственный университет, 2011. – 123 с.
7. Левитин А. В. Алгоритмы. Введение в разработку и анализ / А.В. Левитин – М.: Вильямс, 2006. – 576 с.
8. Немов Я.Н. Модель нарушителя и стратегий его действий в системе физической защиты объекта ФСИН России // Вестник ВИ МВД России. – 2015. – №2. – С. 187-195.
9. Филимоменкова, Т.И. Вариационные задачи в проблеме безопасности и методы их решения / Т.И. Филимоменкова // Вестник ЮУрГУ. – 2011. – Вып. 7. – №4 (221). – С. 111-120.
10. Abakarov A., Sushkov Yu., Almonacid S., Simpson R. Thermal processing optimization through a modified adaptive random search // Journal of Food Engineering. – 2009. – №2. – С. 200-209.
11. Duran, F.A. Probabilistic Basis and Assessment Methodology for Effectiveness of Protecting Nuclear Materials: PhD dissertation / F.A. Duran. – The University of Texas, 2010. – 108 с.
12. Egorov I.N., Kretinin G.V., Leshchenko I.A. Robust Design Optimization Strategy of IOSO Technology // Fifth World Congress on Computational Mechanics / Vienna, Austria., 2002. – С. 8.

13. Jones D. R., Schonlau M., Welch W. J. Efficient Global Optimization of Expensive Black-Box Functions // Journal of Global Optimization. – 1998. – №4. – С. 455-492.
14. Helton J., Davis F. Latin Hypercube Sampling and the Propagation of Uncertainty in Analyses of Complex Systems // Reliability Engineering & System Safety. – 2003. – Vol. 81. – С. 23-69.
15. Holmström K. An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization // Journal of Global Optimization. – 2008. – №3. – С. 447-464.
16. Neubert H., Kamusella A., Pham Th.-Qu. Robust and Reliability-Based Design Optimization of Electromagnetic Actuators Using Heterogeneous Modeling with COMSOL Multiphysics and Dynamic Network Models // 4th European COMSOL Conference / Paris: Comsol, 2010. – С. 100-106.
17. Pintér J. D. Nonlinear optimization with GAMS/LGO // Journal of Global Optimization. – 2008. – №1. – С. 79-101.
18. Regis R., Shoemaker C. Constrained Global Optimization of Expensive Black Box Functions Using Radial Basis Functions // Journal of Global Optimization. – 2005. – №1. – С. 153-171.
19. Sahinidis N. BARON: A general purpose global optimization software package. // Journal of Global Optimization. – 1996. – №8. – С. 201-205.

ПРИЛОЖЕНИЕ 1

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра дифференциальных и стохастических уравнений

ОПТИМИЗАЦИЯ ЭФФЕКТИВНОСТИ СИСТЕМЫ БЕЗОПАСНОСТИ ОХРАНЯЕМОГО ОБЪЕКТА

ТЕКСТ ПРОГРАММЫ
ЮУрГУ – 09.04.04.2016.129-128.ВКР

Руководитель, канд. физ.-мат. наук, доцент
_____ / С.М. Елсаков /
«___» _____2016 г.

Автор, студент группы ММиКН-293
_____ / В.М. Корсаков /
«30» мая 2016 г.

Нормоконтролер, канд. физ.-мат. наук, доцент
_____ / М.А. Сагадеева /
«___» _____2016 г.

Челябинск 2016

УДК – 51-74

Корсаков В.М.

Оптимизация эффективности системы безопасности охраняемого объекта. / В.М. Корсаков. – Челябинск, 2016. – 20 с.

В данном приложении приводится исходный текст модулей программного комплекса. Модуль ModelAssessment написан с использованием языка C++ и среды разработки JetBrains CLion. Модуль ModelOptimization разработан на языке Python в среде разработки JetBrains PyCharm. Модуль ModelVisualizer написан на языке Matlab с использованием пакета GNU Octave.

ОГЛАВЛЕНИЕ

П.1.1 ИСХОДНЫЙ ТЕКСТ МОДУЛЯ MODELASSESSMENT

П1.1.1 Исходный текст файла «Program.cpp»

```
#include <iostream>
#include <queue>
#include <fstream>

#include "Field.h"

std::vector< std::vector< Point > > map;

Field loadFieldFromFile(std::string fileName, std::string objectsName) {

   Field field;

   std::ifstream in(fileName.c_str());
   std::ifstream in2(objectsName.c_str());

   if(in.is_open() && in2.is_open()) {

      int width, height, rows, cols;
      in >> width >> height >> cols >> rows;

      field = Field(width, height, cols, rows);

      int objectsCount;
      in2 >> objectsCount;

      for(int i = 0; i < objectsCount; i++) {
         double x, y, h;
         in2 >> x >> y >> h;
         SecurityTower* tower = new SecurityTower(x, y, h);
         field.AddProtectionObject(tower);
      }

      int targetCount;
      in2 >> targetCount;
      for(int i = 0; i < targetCount; i++) {
         double x, y;
         in2 >> x >> y;
         field.AddTarget(SecurityTarget(x, y));
      }
   }
   else {
      field = Field();
      std::cerr << "Cannot load file " << fileName << "!" << std::endl;
   }

   in.close();
   in2.close();


   return field;

}

int main(int argc, char **argv) {

   Field field = loadFieldFromFile("D:\\files\\field.txt",
"D:\\files\\objects.txt");
   field.ComputeInfiltrationMap();
```

```
    double assesment = 0.0;
    map = field.BFS(assesment);

    std::cout.precision(8);

    std::cout << assesment << std::endl;

    field.PrintMap();
    freopen("D:\\files\\output2.txt", "w", stdout);
    std::cout.precision(6);
    for(int i = 0; i < map.size(); i++)
    {
       for(int j = 0; j < map[i].size(); j++)
       {
          std::cout << map[i][j].value << " ";
       }
       std::cout << std::endl;
    }

    return 0;
}
```

П1.1.2 Исходный текст файла «Field.h»

```
#ifndef FIELD_H_
#define FIELD_H_

#include <vector>
#include <queue>
#include <iostream>
#include <stdio.h>

#include "SecurityObject.h"
#include "SecurityTarget.h"
#include "Point.h"

class Field {

public:

   double Width;
   double Height;
   int Cols;
   int Rows;


   Field() : Field (1.0, 1.0, 1, 1) { }

   Field (int rows, int cols) : Field((double) rows, (double) cols, cols,
rows) {     }

   Field(double width, double height, int cols, int rows)
   {
      Rows = rows;
      Cols = cols;
      Height = height;
      Width = width;
      objects = std::vector<SecurityObject*>();
      targets = std::vector<SecurityTarget>();
      infiltrationMap = std::vector<std::vector<double> >(Rows,
std::vector<double>(Cols, 1.0));
   }

   void AddProtectionObject(SecurityObject *securityObject){
      objects.push_back(securityObject);
   }

   void ClearProtectionObjects(){
      objects.clear();
   }

   void AddTarget(SecurityTarget securityTarget){
      targets.push_back(securityTarget);
   }

   void ClearTargets(){
      targets.clear();
   }

   void ComputeInfiltrationMap();

   std::vector< std::vector< Point > > BFS(double&);

   void PrintMap();
```

```
   void AssessInfiltrationProbability();

private:
   std::vector<SecurityObject*> objects;
   std::vector<SecurityTarget> targets;
   std::vector<std::vector<double> > infiltrationMap;
};

#endif /* FIELD_H_ */
```

П1.1.3 Исходный текст файла «Field.cpp»

```
#include <fstream>
#include "Field.h"
#include "Point.h"

void Field::ComputeInfiltrationMap() {

   double cellWidth = Width / Cols;
   double cellHeight = Height / Rows;
   double d = cellWidth * cellHeight;

   for (int i = 0; i < Rows; i++)
   {
      for (int j = 0; j < Cols; j++)
      {
         infiltrationMap[i][j] = 1.0;
         for (int k = 0; k < objects.size(); k++)
         {
            infiltrationMap[i][j] *= (1 - objects[k]-
>DetectionProbability(i*cellWidth, j*cellHeight));
         }
         for (int k = 0; k < objects.size(); k++)
         {
            if(objects[k]->isBlindEye(i*cellWidth, j*cellHeight))
            {
               infiltrationMap[i][j] = 1;
               break;
            }
         }
         infiltrationMap[i][j] = 1 - (10000*infiltrationMap[i][j] /
Cols*Rows)/10000;
      }
   }
}

std::vector< std::vector< Point > > Field::BFS(double &assesment) {

   const int n = infiltrationMap.size();
   const int m = infiltrationMap[0].size();

   //Количество соседних узлов для просмотра - 4 или 8
   const int connectivity = 4;

   //Смещения для соседних узлов
   const int dX[8] = {0, 0, 1, -1, 1, 1, -1, -1};
   const int dY[8] = {-1, 1, 0, 0, 1, -1, 1, -1};

   //Карта вероятностей проникновения
   std::vector< std::vector< double > > result(n, std::vector<double>(m,
0.0));
   //Карта уже просмотренных узлов
   std::vector< std::vector< Point > > checkedPoints(n, std::vector< Point
>(m, Point()));

   // В очереди храним пару из максимального значения вероятности
проникновения
   // и координаты данной точки
   std::priority_queue<Point> floodFillQueue;

   // Изначально в очередь добавляются точки на границе области
   for(int i = 0; i < n; i++){
      floodFillQueue.push(Point(infiltrationMap[i][0], i, 0));
```

```
        floodFillQueue.push(Point(infiltrationMap[i][m-1], i, m-1));
        checkedPoints[i][0] = Point(infiltrationMap[i][0], i, 0);
        checkedPoints[i][m-1] = Point(infiltrationMap[i][m-1], i, m-1);
    }

    for(int i = 0; i < m; i++){

        floodFillQueue.push(Point(infiltrationMap[0][i], 0, i));

        floodFillQueue.push(Point(infiltrationMap[n-1][i], n-1, i));

        checkedPoints[0][i] = Point(infiltrationMap[0][i], 0, i);

        checkedPoints[n-1][i] = Point(infiltrationMap[n-1][i], n-1, i);
    }

    while(!floodFillQueue.empty()){
        //Извлекаем новый элемент из очереди

        double value = floodFillQueue.top().value;

        int currX = floodFillQueue.top().x;

        int currY = floodFillQueue.top().y;

        floodFillQueue.pop();

        //Рассматриваем соседние узлы
        for(int i = 0; i < connectivity; i++){

            int x = currX + dX[i];
            int y = currY + dY[i];

            //Выход за границы
            if(x < 0 || x >= n || y < 0 || y >= m)
                continue;
            double infV = infiltrationMap[x][y];
            double newValue = value + infV;
            double chV = checkedPoints[x][y].value;

            // Точка уже была рассмотрена и уже найденная
            // вероятность обнаружения в ней ниже
            if(checkedPoints[x][y].value <= newValue){
                continue;
            }
            //Добавляем новую точку в очередь и помечаем как рассмотренную
            floodFillQueue.push(Point(newValue, x, y));
            checkedPoints[x][y] = Point(newValue, currX, currY);
        }
    }
    assesment = std::min(std::min(checkedPoints[n/2-11][m/2-11].value,
checkedPoints[n/2-11][m/2+10].value),
                    std::min(checkedPoints[n/2+10][m/2-11].value,
checkedPoints[n/2+10][m/2+10].value));
    return checkedPoints;
}

void Field::PrintMap()
{

    std::ofstream of("D:\\files\\output.txt");
```

```
        of.setf(std::ios::fixed);
        of.precision(8);
        for (int i = 0; i < Rows; i++)
        {
           for(int j = 0; j < Cols; j++)
           {
              of << infiltrationMap[i][j] << " ";
           }
           of << std::endl;
        }
     }
```

П1.1.4 Исходный текст файла «Point.h»

```
#ifndef POINT_H_
#define POINT_H_

#include <limits>

class Point {
public:
   double value;
   int x;
   int y;
   Point() : value(std::numeric_limits<int>::max()), x(0), y(0) { };
   Point(double V, int X, int Y) : value(V), x(X), y(Y) { };
   bool operator < (const Point &arg) const {
      return this->value > arg.value;
   };
};

#endif /* POINT_H_ */
```

П1.1.6 Исходный текст файла «SecurityObject.h»

```
#ifndef SECURITYOBJECT_H_
#define SECURITYOBJECT_H_

#include <cmath>
#include <iostream>

static double minR = 3;
static double maxR = 9;
static double cR = (maxR+minR)/2;
static double maxdR = (maxR-minR)/2;
static double maxPhi = 0.5;

class SecurityObject {

public:

   double X;
   double Y;

   SecurityObject() : X(0), Y(0) { }

   SecurityObject(double x, double y) : X(x), Y(y) { }

   virtual double DetectionProbability(double x, double y) { return 0.0; } ;

   virtual bool isBlindEye(double x, double y) { return true; } ;

   virtual ~SecurityObject() { }

};

class SecurityCamera : public SecurityObject {
public:

   double Direction;

   SecurityCamera() : SecurityObject() {
      Direction = 1;
   }

   SecurityCamera(double x, double y, double direction = 1.0) :
SecurityObject(x, y) {
      Direction = direction;
   }

   double DetectionProbability(double x, double y)    {
      if(X - x < 0.00001)
      {
         return 0;
      }
      double dX = X - x;
      double dY = Y - y;
      double r = fabs(cR-sqrt(dX*dX + dY*dY));
      double phi = fabs(Direction - atan2(dY, dX));

      if ( r > maxdR || phi > maxPhi)
      {
         return 0.0;
      }
      else
      {
```

```
            return ((maxdR-r)/(maxdR))*((maxPhi-phi)/(maxPhi));
         }
      }

      bool isBlindEye(double x, double y)
      {
         double dX = X - x;
         double dY = Y - y;
         double D = sqrt(dX*dX + dY*dY);

         return D <= 1.5;
      }
   };
   class SecurityTower : public SecurityObject {
   public:
      double Height;

      SecurityTower() : SecurityObject() {
         Height = 1;
      }

      SecurityTower(double x, double y, double height = 1.0) : SecurityObject(x,
   y) {
         Height = height;
      }

      double DetectionProbability(double x, double y)     {

   //    if(X > 12.5 && X < 17.5 && X > 12.5 && X < 17.5)
   //    {
   //       return 0.0;
   //    }

         double dX = X - x;
         double dY = Y - y;
         double D = sqrt(dX*dX + dY*dY);

         double c = 3.1415 * Height * Height / 3;

         if (D > Height || D < Height / 4)
         {
            return 0.0;
         }
         else
         {
            return ((Height - D)/Height);
         }
      }

      bool isBlindEye(double x, double y)
      {
         double dX = X - x;
         double dY = Y - y;
         double D = sqrt(dX*dX + dY*dY);

         return D <= 1.5;
      }

   };

   #endif /* SECURITYOBJECT_H_ */
```

П1.1.7 Исходный текст файла «SecurityTarget.cpp»

```
#ifndef SECURITYTARGET_H_
#define SECURITYTARGET_H_

class SecurityTarget {
public:

   double X;
   double Y;

   SecurityTarget() : SecurityTarget(0,0) { }

   SecurityTarget(double x, double y) {
      X = x;
      Y = y;
   }
};

#endif /* SECURITYTARGET_H_ */
```

П1.2 ИСХОДНЫЙ ТЕКСТ МОДУЛЯ MODELVISUALIZER

```
%Входные параметры:
%X - значение координаты X в узлах сетки
%Y - значение координаты Y в узлах сетки
%modelInput - имя файла для считывания рассчитанных значений функции обнаружения
%modelInput - имя файла для считывания рассчитанных значений оценки вероятности
%             проникновения
function [] = ModelVisualizer(X, Y, modelInput, assessmentInput)

  % Считывание матрицы с моделью из файла
  M1 = dlmread(modelInput);
  % Выделение нового окна для графика
  figure(1)
  % Отображение графика
  surfc(X, Y, M1)

  % Считывание матрицы с оценкой из файла
  M2 = dlmread(assessmentInput);
  % Выделение нового окна для графика
  figure(2)
  % Отображение графика
  surfc(X, Y, M2)
```

П1.3 ИСХОДНЫЙ ТЕКСТ МОДУЛЯ MODELOPTIMIZATION

П1.3.1 Исходный текст файла «RBFInterpolation.py»

```
def rbf(pts, T):

    n = len(pts)
    d = len(pts[0]) - 1

    def phi(r):
        return r * r * r

    Phi = np.zeros((n, n))
    for i in range(n):

        for j in range(n):

            Phi[i, j] = phi(np.linalg.norm(np.dot(T, np.subtract(pts[i, 0:-1],
pts[j, 0:-1]))))

    P = np.ones((n, d + 1))

    for i in range(n):

        P[i, 0:-1] = pts[i, 0:-1]

    F = np.zeros(n)

    for i in range(n):

        F[i] = pts[i, -1]

    M = np.zeros((n + d + 1, n + d + 1))
    M[0:n, 0:n] = Phi
    M[0:n, n:n + d + 1] = P
    M[n:n + d + 1, 0:n] = np.transpose(P)

    v = np.zeros(n + d + 1)
    v[0:n] = F

    sol = np.linalg.solve(M, v)

    lam = sol[0:n]

    b = sol[n:n + d]

    a = sol[n + d]


    def fit(z):
        res = 0.

          for i in range(n):
```

```
            res = res + lam[i] * phi(np.linalg.norm(np.dot(T, np.subtract(z,
    pts[i, 0:-1]))))
        res = res + np.dot(b, z) + a
        return res

    return fit
```

П1.3.2 Исходный текст файла «LatinHypercube.py»

```
def latin(n, d):

      pts = np.ones((n, d))

    for i in range(n):
        pts[i] = pts[i] * i / (n - 1.)

    def spread(p):
        s = 0.
        for i in range(n):
            for j in range(n):
                if i > j:
                    s = s + 1. / np.linalg.norm(np.subtract(p[i], p[j]))
        return s

    currminspread = spread(pts)

    for m in range(1000):

        p1 = np.random.randint(n)
        p2 = np.random.randint(n)
        k = np.random.randint(d)

        newpts = np.copy(pts)
        newpts[p1, k], newpts[p2, k] = newpts[p2, k], newpts[p1, k]
        newspread = spread(newpts)

        if newspread < currminspread:
            pts = np.copy(newpts)
            currminspread = newspread

    return pts
```

П1.3.2 Исходный текст файла «Optimization.py»

```
import numpy as np
import csv
import multiprocessing as mp
import math
import subprocess

def search(f, resfile, box, cores, n, it, tratio=0.75, rho0=0.75, p=0.75,
nrand=10000, vf=0.05):

    d = len(box)

    if np.mod(n, cores) != 0:
        n = n - np.mod(n, cores) + cores

    if np.mod(it, cores) != 0:
        it = it - np.mod(it, cores) + cores

    def cubetobox(pt):
        res = np.zeros(d)
        for i in range(d):
            res[i] = box[i, 0] + (box[i, 1] - box[i, 0]) * pt[i]
        return res

    def cubetoboxFinal(pt):
        res = np.zeros(d+1)
        for i in range(d):
            res[i] = box[i, 0] + (box[i, 1] - box[i, 0]) * pt[i]
        res[d] = pt[d]
        return res

    pts = np.zeros((n, d + 1))
    lh = latin(n, d)

    for i in range(n):
        for j in range(d):
            pts[i, j] = lh[i, j]

    for i in range((math.floor(n / cores))):
        pts[cores * i:cores * (i + 1), -1] = pmap(f, map(cubetobox, pts[cores
* i:cores * (i + 1), 0:-1]), cores)

    t = pts[pts[:, -1].argsort()][np.ceil(tratio * n) - 1, -1]

    def fscale(fval):
        if fval < t:
            return fval / t
        else:
            return 1.

    for i in range(n):
        pts[i, -1] = fscale(pts[i, -1])

    if np.mod(d, 2) == 0:
```

```
        vl = np.pi ** (d / 2) / np.math.factorial(d / 2)
    else:
        vl = 2 * (4 * np.pi) ** ((d - 1) / 2) * np.math.factorial((d - 1) / 2)
/ np.math.factorial(d)

    T = np.identity(d)

    for h in range(math.floor(it / cores)):

        if d > 1:

            pcafit = rbf(pts, np.identity(d))

            cover = np.zeros((nrand, d + 1))
            cover[:, 0:-1] = np.random.rand(nrand, d)
            for i in range(nrand):
                cover[i, -1] = pcafit(cover[i, 0:-1])

            cloud = cover[cover[:, -1].argsort()][0:np.ceil(vf * nrand), 0:-1]

            eigval, eigvec = np.linalg.eig(np.cov(np.transpose(cloud)))

            T = np.zeros((d, d))
            for i in range(d):
                T[i] = eigvec[:, i] / np.sqrt(eigval[i])
            T = T / np.linalg.norm(T)

        fit = rbf(pts, T)

        pts = np.append(pts, np.zeros((cores, d + 1)), axis=0)

        for i in range(cores):

            r = ((rho0 * ((it - 1. - (h * cores + i)) / (it - 1.)) ** p) / (vl
* (n + (h * cores + i)))) ** (1. / d)

            fitmin = 1.
            for j in range(nrand):

                x = np.random.rand(d)
                ok = True

                if fit(x) < fitmin:

                    for k in range(n + h * cores + i):
                        if np.linalg.norm(np.subtract(x, pts[k, 0:-1])) < r:
                            ok = False
                            break
                else:
                    ok = False

                if ok == True:
                    pts[n + h * cores + i, 0:-1] = np.copy(x)
                    fitmin = fit(x)
```

```
        smth = pmap(f, map(cubetobox,pts[n + cores * h:n + cores * (h + 1), 0:-
1]), cores)
        pts[n + cores * h:n + cores * (h + 1), -1] = list(map(fscale, smth))

    extfile = open(resfile, 'w')
    wr = csv.writer(extfile, dialect='excel')
    for item in pts:
        wr.writerow(cubetoboxFinal(item))

    extfile = open(output, 'w')
    wr = csv.writer(extfile, dialect='excel')
    for item in pts:
        wr.writerow(item)
```