УДК 004.896

Применение бионических моделей и методов для решения оптимизационных задач проектирования агротехнических систем полива

В. Д. Фроловский, Л. Ю. Забелин, С. Л. Забелин

В работе реализованы и исследованы приближенные метаэвристические алгоритмы решения задачи покрытия ортогональной области с запретами кругами разного радиуса с использованием бионических принципов. Проведены вычислительные эксперименты как на случайно сгенерированных объектах, так и на моделях реальных проектов. Проанализированы результаты, полученные в ходе экспериментов. Сделаны выводы об эффективности методов и их использовании в реальных задачах.

Ключевые слова: геометрические покрытия, оптимизационная задача, бионические методы, размещение устройств полива.

1. Введение

Задача геометрического покрытия относится к проблематике «раскроя и упаковки» (*Cutting and Packaging, C&P*). Имеется некоторая поверхность, которую нужно покрыть ограниченным количеством объектов, размер которых может отличаться. Важно, чтобы была покрыта вся заданная область и при этом было как можно меньше перекрытий. Актуальность задачи геометрического покрытия обусловлена ее принадлежностью к классу *NP*-трудных задач, причем с дискретно-непрерывной структурой. Поэтому возникает проблема разработки приближенных и эвристических методов, позволяющих находить субоптимальные решения. [1]. Область применения описанной задачи велика и включает системы безопасности, противопожарные системы, агротехнические системы полива, воздушное и космическое наблюдение, химию, разработку компьютерных игр и прочие.

На данный момент многие компании занимаются проектированием систем автоматического полива, такие как Лэнд Строй [2], Автополив-Сиб [3], *Indiga* [4], Аквабаланс [5] и др. Как правило, количество и расположение устройств полива приводит к большой площади перекрытия этих устройств и удорожанию системы. Зачастую решение задачи выполняется вручную, что неизбежно ведет к значительным временным затратам.

Точные методы решения, в свою очередь, имеют экспоненциальную сложность и не позволяют получить решение за приемлемое время. Применение оптимизационных методов на основе бионических принципов – одно из актуальных и перспективных направлений.

2. Постановка задачи

Для решения задачи геометрического покрытия в рамках данной работы выбраны алгоритм роя частиц, пчелиный алгоритм и алгоритм отжига.

Входными данными является набор $S=\langle w,h,R,O\rangle$, где w – ширина области A, h – длина области A, R – набор радиусов покрывающих объектов, O – множество запретных зон.

Для визуализации области вокруг покрываемой поверхности вводится дополнительное запретное пространство длиной, равной максимальному радиусу покрывающих объектов. Это позволит полностью отображать объекты в случае их выхода за границу вне зависимости от их радиуса.

Множество запретных зон $O=\langle O_1,\dots O_n\rangle$, где $O_i=\langle S_x,S_y,O_h,O_w\rangle$, где S_x – координата X левой верхней вершины запретной зоны, S_y – координата Y левой верхней вершины запретной зоны, O_w – ширина запретной зоны, O_h – длина запретной зоны.

Стороны запретных зон параллельны сторонам покрываемой поверхности, варианты с поворотом запретных зон не рассматриваются в рамках данной работы. Требуется покрыть конечным количеством кругов радиусами из набора R площадь A/O.

Выходными данными является набор $Z=\langle C,F,T,N\rangle$, где C – покрывающие круги $C=\langle r,C_X,C_Y\rangle$, r – радиус, C_X – координата X центра круга, C_Y – координата Y центра круга, F – значение целевой функции, T – время выполнения алгоритма, N – количество покрывающих объектов.

В качестве целевой функции можно использовать следующее выражение:

$$F=K_s-K_M-K_O,$$

где $F\to 1$, K_S – коэффициент покрытия поверхности, K_M – коэффициент, показывающий отношение площади покрывающих кругов, вышедших за границы области или попавших на запретные зоны, к общей площади всех кругов, K_I – коэффициент, показывающий отношение площади перекрытий кругов к общей площади всех кругов.

Опишем условия, при которых решение считается допустимым. Будем рассматривать многосвязные ограниченные полигоны (МОП – ортогональный полигон с прямоугольными запретными областями). Должна быть покрыта вся поверхность, т.е. для любой точки МОП найдётся объект, покрывающий её.

$$\forall P\in A/Z\quad \exists C_i\in C: P\in C_i.$$

Центры всех покрывающих объектов не выходят за границы МОП:

$$\forall C_i\in C, C_{x_i}\in[0;W], C_{Y_i}\in[0;H], i=\overline{1,N}.$$

Центры всех покрывающих объектов не находятся внутри запретных зон:

$$\forall C_i\in C, C_i\cap O_j=\varnothing,\ i=\overline{1,N}, j=\overline{1,M}.$$

Значение целевой функции покажет оптимальность решения и эффективность выбранного метода. Алгоритмы отжига, частиц и пчел используют некое начальное приближение в качестве первого шага. Для выполнения этого шага можно использовать случайное заполнение МОП объектами или иные алгоритмы.

3. Методы решения

3.1. Алгоритм «первый подходящий»

Первый подходящий алгоритм (*first fit*) предполагает простую укладку первого попавшегося геометрического объекта на первую найденную пустую область на покрываемой поверхности. Покрытие происходит слева направо, сверху вниз. Он не гарантирует получения хорошего результата, но даёт возможность построения хорошего начального приближения.

Этапы работы алгоритма:

1. Случайный выбор покрывающего объекта из списка доступных объектов. В рамках данной задачи это подразумевает выбор радиуса r объекта из списка радиусов R.

2. Последовательный поиск первой непокрытой точки на МОП P, такой, что покрывающий объект полностью входит в МОП и не перекрывает ни одну запретную зону.

3. Если такой точки нет, то выбирается первая свободная точка.

4. Покрытые объектом точки помечаются, чтобы не учитывать их на последующих итерациях. Алгоритм повторяется, пока не будут покрыты все точки поверхности.

3.2. Жадный алгоритм

Принцип жадности подразумевает выбор оптимального решения на каждом шаге с предположением, что это приведет к глобальному оптимальному решению.

Несмотря на пригодность для широкого класса задач, жадные алгоритмы не всегда дают оптимальное решение, но могут дать допустимое. Также известно, что для некоторых *NP*-задач жадные алгоритмы оптимального решения не дают.

В рамках задачи покрытия МОП кругами жадный алгоритм можно представить в виде следующих этапов.

1. Случайный выбор покрывающего объекта из списка доступных объектов. В рамках данной задачи это подразумевает выбор радиуса r объекта из списка радиусов R.

2. Выбор точки, наиболее отстоящей от всех запретных зон и всех ранее размещенных покрывающих объектов.

Этапы повторяются до тех пор, пока не будет покрыта вся поверхность.

Второй этап выполняется путем последовательной проверки всех свободных точек и расчета расстояний от них до каждого покрывающего объекта (его центра) и каждой запретной зоны. Среди этих расстояний выбирается минимальное. Среди всех выбранных минимумов берется с максимальным значением. Таким образом, решается задача минимакса на каждой итерации алгоритма.

В рамках данной работы принцип жадности совмещен со случайным размещением объектов. Первые 50 % покрываемой поверхности покрываются случайным образом так, чтобы покрывающий объект полностью лежал внутри МОП и не пересекался с запретными зонами. Затем для оставшейся поверхности используется описанный жадный алгоритм.

Возможна модификация этого метода, включающая следующие этапы.

1. Случайный выбор покрывающего объекта из списка доступных объектов. В рамках данной задачи это подразумевает выбор радиуса r объекта из списка радиусов R.

2. Выбор точки, наиболее отстоящей от всех запретных зон и всех ранее размещенных покрывающих объектов, причём такой, чтобы покрывающий объект полностью располагался внутри покрываемой области.

3. Если такой точки нет, последнее ограничение из шага 2 снимается и выбирается точка, наиболее удаленная ото всех объектов и запретных зон.

3.3. Алгоритм имитации отжига

Алгоритм имитации отжига возник в середине 1980-х годов и основан на аналогии с процессом кристаллизации вещества, например, при отжиге металла. В ходе этого процесса температура вещества понижается, оно отвердевает, при этом замедляется скорость движения частиц вещества [6–8]. У каждого металла есть кристаллическая решетка. Совокупность позиций всех атомов будем называть состоянием системы, каждому состоянию соответствует определенный уровень энергии. Цель отжига – привести систему в состояние с наименьшей энергией. Чем ниже уровень энергии, тем меньше у нее дефектов и прочнее металл. В ходе отжига металл сначала нагревают до некоторой температуры, что заставляет атомы кристал-

лической решетки покинуть свои позиции. Затем начинается медленное и контролируемое охлаждение. Атомы стремятся попасть в состояние с меньшей энергией, однако с определенной вероятностью они могут перейти и в состояние с большей. Эта вероятность уменьшается вместе с температурой. Процесс завершается, когда температура падает до заранее заданного значения [7, 8].

В качестве начального решения можно предложить решение, полученное другими методами, например, с помощью жадного алгоритма, описанного ранее. Единственным требованием является получение в качестве оценки одного вещественного числа, которое будет характеризовать оптимальность предлагаемого решения, т.е. энергии. В качестве неё будет выступать значение целевой функции. Выбор способа уменьшения температуры может быть различным и выбирается экспериментально. Главное, чтобы температура монотонно убывала к нулю. Хорошей стратегий является умножение на каждом шаге температуры на некоторый коэффициент, меньший единицы. Выбор границ температуры тоже производится экспериментальным путем.

Опишем алгоритм в рамках задачи геометрического покрытия.

1. Генерируется случайное состояние, т.е. координаты всех покрывающих объектов, и рассчитывается значение функции. Если оно больше наилучшего (функция стремится к 1), то принимаем его за наилучшее.

2. Вычисляются новые координаты объектов по функции $x = x + r_1 M_x \ln r_2$, где x – одна из координат объекта, r_1 – случайное число (-1 или +1), r_2 – случайное число, равномерно распределенное от 0 до 1, M_X – параметр алгоритма, от которого зависит, как далеко от текущей точки может переместиться процесс поиска за один шаг. Если новое состояние лучше текущего, осуществляется переход в него, иначе переход определяется функцией:

$$\frac{\varphi(TX_{\text{new}}) - \varphi(TX)}{T} > r,$$

где T – текущая температура, r – случайное число, равномерно распределённое от 0 до 1. Новое значение температуры рассчитывается по формуле:

$$T = \frac{T_o}{e^{i*v}},$$

где T_0 – начальная температура, а v – коэффициент снижения температуры.

Алгоритм завершает свою работу по достижении конечной (минимальной) температуры.

3.4. Алгоритм роя частиц

В 1995 году Джеймс Кеннеди и Рассел Эберхарт предложили метод для оптимизации непрерывных нелинейных функций, названный ими алгоритмом роя частиц. Он моделирует многоагентную систему, где агенты-частицы двигаются к оптимальным решениям, обмениваясь информацией с соседями [7, 8]. Стая птиц всегда действует скоординированно, каждая птица действует согласно простым правилам, следит за другими птицами и согласует свое движение с ними. Найдя источник пищи, птица сообщает о нем всей стае. Источники пищи обычно расположены случайным образом и одной птице очень сложно быстро найти их. Текущее состояние частицы характеризуется координатами в пространстве решений (то есть, собственно, связанным с ними решением), а также вектором скорости перемещения. Оба этих параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит координаты лучшего из найденных ей решений, а также лучшее из пройденных всеми частицами решений – этим имитируется мгновенный обмен информацией между птицами [7, 8]. Основная идея метода заключается в перемещении частиц в пространстве решений. Пусть решается задача нахождения минимума (максимума) функции вида $f(X)$, где X – вектор варьируемых параметров, которые могут принимать значения из не-

которой области D. Тогда каждая частица в каждый момент времени характеризуется значением параметров X из области D (координатами точки в пространстве решений) и значением оптимизируемой функции $f(X)$ (привлекательностью данной точки). При этом частица «помнит» наилучшую точку в пространстве решений, в которой была, и стремиться в нее вернуться, но подчиняется также закону инерции и имеет склонность к небольшому стохастическому изменению направления движения. Однако этих правил недостаточно для перехода к системе, так как не заданы связи между элементами. В качестве связи используется так называемая общая память, благодаря которой каждая частица знает координаты наилучшей точки среди всех, в которых была любая частица роя. В итоге на движение частицы влияют стремление к своему наилучшему положению, стремление к наилучшему среди всех частиц положению, инерционность и случайные отклонения. Алгоритм завершается при достижении заданного числа итераций, либо при достижении удовлетворительного решения, либо после истечения отведенного на работу времени. Основные этапы алгоритма.

1. Случайно распределить частицы в области решения, назначить нулевые начальные скорости.

2. Вычислить значения оптимизируемой функции по каждой частице с обновлением при необходимости локальных и глобальных лучших решений.

3. Вычислить новые значения скоростей по каждой частице.

4. Вычислить новые координаты частиц.

5. Если выполнено условие завершения, закончить алгоритм, иначе – перейти на шаг 2.

В рамках задачи геометрического покрытия частицами будем считать решения задачи. Координатами являются все координаты всех покрывающих объектов. Лучшее локальное и лучшее глобальное решения будут представлять значения целевых функций.

Алгоритм роя частиц можно представить в следующем виде.

1. Получить множество частиц $S = \{S_1, S_2, \ldots, S_N\}$.

2. Рассчитать все значения целевых функций F.

3. Если значение F_i лучше лучшего локального F_{i_b}, обновить локальное наилучшее.

4. Если среди наилучших локальных значений F_{i_b} есть большее наилучшего глобального, обновить наилучшее глобальное F_{ult}.

5. Рассчитать скорость движения частиц по формуле:

$$V_{ij+1} = V_{ij}\omega + \alpha_1\left(X_{ij}^{\text{best}} - X_{ij}\right)\text{rnd}_1 + \alpha_2\left(M - X_{ij}\right)\text{rnd}_2,\ \ i = 1, \ldots, N,$$

где rnd_1 и rnd_2 – случайные числа, равномерно распределенные в интервале, X – положение частицы, коэффициенты a_1 и a_2 определяют степень учета индивидуального и группового опыта частиц соответственно, а ω характеризует инерционные свойства.

6. Пересчитать параметры (координаты) по формулам

$$X_{ij+1} = \begin{cases} X_{ij} + V_{ij+1}, G(X_{ij} + V_{ij+1}) = 1 \\ X_{ij}, G(X_{ij} + V_{ij+1}) = 0 \end{cases}, i = 1, \ldots, N,$$

где $G(X)$ – предикат, который показывает, принадлежит ли X области допустимых значений.

7. Повторить итерацию. Алгоритм завершается по истечении заданного числа итераций или времени.

3.5. Алгоритм роя пчел

Алгоритм роя пчел (*Artificial Bee Colony Algorithm* или *Bees Algorithm*) разработан в 2005 году [8]. Метод основан на симуляции поведения пчел при поиске нектара. Рой пчел отправляет несколько разведчиков в случайных направлениях для поиска нектара. Вернувшись,

разведчики сообщают о найденных на поле участках с цветами, содержащими нектар, и на них вылетают остальные пчелы. При этом чем больше на участке нектара, тем больше пчел к нему устремляется. Однако при этом пчелы могут случайным образом отклоняться от выбранного направления. После возвращения всех пчел в улей вновь происходит обмен информацией и отправка пчел-разведчиков и пчел-рабочих. Фактически разведчики действуют по алгоритму случайного поиска.

Частица, или агент, – каждая пчела в рое. Все частицы роя действуют индивидуально в соответствии с одним управляющим принципом: ускоряться в направлении наилучшей персональной и наилучшей общей позиции, постоянно проверяя значение текущей позиции.

Позиция – аналогично местоположению пчелы на поле представляется координатами на плоскости $\{x; y\}$. Однако в общем случае можно расширить эту идею на любое *N*-мерное пространство в соответствии с поставленной задачей. Это *N*-мерное пространство является областью решений для оптимизируемой задачи, где каждый набор координат представляет решение.

Пригодность – по аналогии с примером пчелиного роя функция пригодности будет плотностью цветов: чем больше плотность, тем лучше позиция. Функция пригодности служит средством связи между физической проблемой и алгоритмом оптимизации.

Персональная наилучшая позиция (ПНП) – по аналогии с пчелиным роем, каждая пчела помнит позицию, где она сама обнаружила наибольшее количество цветов. Эта позиция с наибольшим значением пригодности, обнаруженная пчелой, известна как персональная наилучшая позиция. Каждая пчела имеет собственное значение ПНП, определяемое путем, который она пролетела. В каждой точке вдоль пути движения пчела сравнивает значение пригодности текущей позиции со значением ПНП. Если текущая позиция имеет значение пригодности выше, значение ПНП заменяется на значение текущей позиции.

Глобальная наилучшая позиция – каждая пчела также каким-то образом узнает область наибольшей концентрации цветов, определенную всем роем. Эта позиция наибольшей пригодности известна как глобальная наилучшая позиция (ГНП). Для всего роя это одна ГНП, к которой стремится каждая пчела. В каждой точке на протяжении всего пути каждая пчела сравнивает пригодность ее текущей позиции с ГНП. В случае если какая-либо пчела обнаружит позицию с более высокой пригодностью, ГНП заменяется текущей позицией этой пчелы [8, 9]. На каждом шаге работы алгоритма среди всех агентов выбирается n^b лучших по значению целевой функции. Среди прочих выбирается еще n^g лучших, так называемых «выбранных» или «перспективных» (здесь верхние индексы не являются показателем степени).

В некоторых вариантах алгоритма требуется, чтобы расстояния между каждой парой позиций в объединенном множестве лучших и выбранных позиций не превышали определенной величины. Иными словами, если есть две близкие позиции, то худшая из них по значению целевой функции отбрасывается, вместо нее берется позиция другого агента, подходящая под условия.

Определенные таким образом позиции (множество лучших N^b и множество выбранных N^g) запоминаются и на следующем шаге в окрестность каждой лучшей позиции высылается c^b пчел, а каждой выбранной – c^g. Пчела, посылаемая в окрестность участка, попадает в случайную точку внутри окрестности, например, в двумерном пространстве окрестность позиции с центром в точке (x, y) представляет область $([x - rx; x + rx], [y - ry; y + ry])$, где rx и ry являются параметрами алгоритма. Возможно использование одного коэффициента rx по всем измерениям или вектора RX. Пчелы-разведчики в количестве n^s высылаются в случайные позиции по всему пространству поиска на каждой итерации.

Алгоритм завершается при достижении заданного числа итераций, либо при достижении удовлетворительного решения, либо по истечении отведенного на работу времени.

При формализации алгоритма полем считается область решений, а нектаром – целевая функция. На каждом этапе посылается n^s разведчиков, рассчитываются значения целевой функции и выбираются N^b наилучших решений и N^g возможных решений. Если суммарно решений меньше, чем $N^b + N^g$, то берутся все решения. В окрестности этих решений посылаются c^b и c^g пчёл. Окрестность задаётся шириной границы *r*, например $[x - rx; x + rx]$.

Этапы алгоритма представлены ниже.

1. Расчет разведчиков (начальных решений)

$$X_i = \text{rand}(G(X)), i = 1, ..., n^s,$$

где n^s – количество пчел-разведчиков, X_i – вектор координат покрывающих объектов, $G(X)$ – рабочее пространство, т.е. МОП.

2. Вычисление целевой функций каждого разведчика на текущей итерации $F(X_{ij})$.

3. Выбор наилучших и перспективных решений $\left(N_{ij}^b, Ng_{kj}^g\right)$.

4. В окрестность каждой лучшей позиции отправляется c^b пчел, в окрестность каждой перспективной – c^g пчел. В некоторых вариантах алгоритма число отправляющихся в окрестность участка пчел зависит от его качества с точки зрения целевой функции, но в данном описании это не используется. Таким образом, позиции всех пчел-рабочих определяются следующим образом:

$$X_{(i-1)c^b + k\ j} = N_{ij-1}^b + \text{Rnd} * dx, i = 1, ..., n^b, k = 1, ..., c^b,$$

$$X_{n^b c^b + (i-1)c^b + k\ j} = N_{ij-1}^g + \text{Rnd} * dx, i = 1, ..., n^g, k = 1, ..., c^g,$$

где Rnd – вектор, состоящий из l равномерно распределенных случайных величин в диапазоне $(-1, 1)$.

5. Выбор наилучшего значения *F*. Если оно больше глобального, то перезаписываем его. Глобальное решение будет решением задачи. Алгоритм останавливается по окончании итераций или истечении времени.

4. Результаты вычислительных экспериментов и их анализ

Для оценки эффективности алгоритмов было разработано программное обеспечение в среда *Microsoft Visual Studio*, язык программирования – *C#*. Проведены вычислительные эксперименты на 12 модельных и 3 реальных задачах. В модельных задачах выполнялась генерация объектов со случайным расположением и размерами препятствий. В качестве генератора псевдослучайных чисел использовался системный класс *Random* языка программирования *C#*.

При помощи генератора псевдослучайных чисел формировались препятствия случайных размеров. Накладываются ограничения на их размеры и расположение на плоскости. Для представления покрывающих объектов, используемых для поиска решения, формировались 3 параметра: радиусы и координаты центра окружностей. Эксперименты проводились на 64-разрядном компьютере с тактовой частотой процессора 2 ГГц и объемом оперативной памяти 8 ГБ. Значения времени решения и целевой функции усреднялись по значениям выборки объектов.

Сравнительные результаты работы алгоритмов представлены в табл. 1.

Таблица 1. Результаты экспериментов на модельных задачах

Алгоритм	Время, с	Функция	Число объектов
Жадный	2.57	0.378	40
Первый подходящий	1.94	0.362	48
Пчелиный	326.7	0.391	41
Метод частиц	34.7	0.387	44
Метод имитации отжига	20.6	0.401	48

Эксперименты проводились на объектах, сформированных на основе реальных проектов [2–5]. В качестве примера на рис. 1 и в табл. 2 отражены результаты для реальных объектов проекта «Мартемьяново» [5], покрываемая площадь которого – 23 сотки.

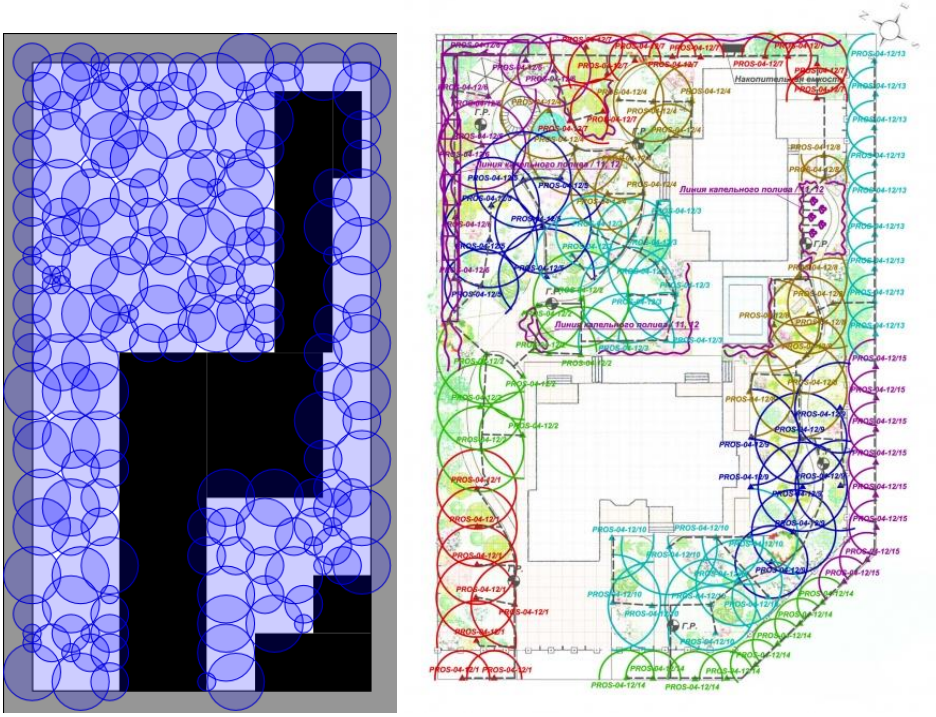

Рис. 1. Проект «Мартемьяново»

Таблица 2. Результаты по проекту «Мартемьяново»

Алгоритм	Время, с	Функция	Число объектов
Жадный	0.54	0.622	85
Первый подходящий	0.36	0.606	93
Пчелиный	37.5	0.658	84
Метод частиц	4.15	0.631	78
Метод имитации отжига	0.65	0.620	88
Ручное размещение	до нескольких дней	0.372	79

На случайно сгенерированных объектах без препятствий лучший результат показал алгоритм отжига, за меньшее время нашедший лучшие решения в среднем по выборке. Близкие результаты показывал алгоритм частиц, однако он уступал по времени выполнения. Жадный алгоритм давал лучшие начальные приближения с небольшой разницей во времени в сравнении с первым подходящим. На случайно сгенерированных объектах с препятствиями жадный алгоритм вновь давал лучшее начальное приближение, нежели первый подходящий. Алгоритм отжига показал лучшие результаты как по значению функции, так и по времени среди

двух других. На объектах, сгенерированных пользователем, чаще других лучшее решение показывал пчелиный алгоритм, который имел время выполнения, значительно превышающее время других алгоритмов. Алгоритм частиц показывал средние результаты по времени и эффективности поиска решения. Алгоритм отжига давал за меньшее время значения, равные или большие, чем алгоритм частиц. Жадный алгоритм преимущественно давал лучшие результаты, чем первый подходящий. Ожидаемо, что итерационные алгоритмы требуют больших временных затрат на поиск решения, но они, как правило, дают лучшее решение. Средний прирост значения целевой функции от начального приближения – 3.5 %.

На моделях реальных объектов жадный алгоритм показал существенный прирост значения функции на фоне первого подходящего. Пчелиный алгоритм давал лучшие результаты, но за большее время. Алгоритм отжига за короткое время давал результаты, близкие к алгоритму частиц. При этом алгоритм частиц в большинстве случаев находил решение с меньшим количеством покрывающих объектов.

Среди двух алгоритмов, применяемых в качестве начального приближения, жадный чаще давал лучшее значение функции с меньшим количеством использованных покрывающих объектов, чем первый подходящий, с незначительной разницей во времени. Однако соотношение значений целевой функции между этими двумя алгоритмами говорит о необходимости дальнейших модификаций предложенного жадного алгоритма и поиска конфигурации, дающей больший прирост к значению целевой функции. В частности, можно ввести учёт полного расположения покрывающего объекта внутри области всегда, когда это возможно, а также уход от случайного расположения первых объектов и использование поиска наиболее удаленной точки с первого шага.

На основании полученной статистки можно выделить предложенный вариант жадного алгоритма, а также алгоритмы отжига и частиц. Необходима их дальнейшая модификация, повышение качества начального приближения, варьирование параметров алгоритма отжига для достижения лучших результатов, а также распараллеливание вычислений пчелиного алгоритма (например, его итераций) для значительного повышения производительности и поиска решений на большем количестве агентов и итераций.

5. Заключение

Выполнено сравнение результатов, полученных при помощи алгоритмов роя пчёл, частиц и отжига, по времени и значению предложенной целевой функции как на выборках из случайно сгенерированных объектов с произвольным расположением препятствий, так и на сформированных пользователем, в том числе реально существующих объектах. Результаты показали перспективу дальнейшего развития алгоритма отжига и частиц с целью получения лучших результатов. Создано программное обеспечение, которое позволяет тестировать алгоритмы на случайных выборках, а также на создаваемых пользователем объектах, собирать средние значения по выборке, представлять результаты в графическом виде и экспортировать их, настраивать параметры алгоритмов, такие как число итераций, количество агентов, коэффициенты. Также можно организовать самообучение. Общим подходом для повышения быстродействия можно назвать распараллеливание созданных методов.

Литература

1. *Романовский И. В.* Субоптимальные решения. Петрозаводск: Изд-во ПГУ, 1998. 96 с.
2. Лэнд Строй. Системы автоматического полива. URL: `https://www.landstroy-nsk.ru` (дата обращения 06.06.2018).
3. Автополив-Сиб. Автополив, мощение, дренаж в Новосибирске. URL: `http://avtopoliv-sib.ru` (дата обращения 06.06.2018).

4. *Indiga*. Полив без хлопот. URL: https://www.avtopoliv.me (дата обращения 06.06.2018).
5. Аквабаланс [сайт 2003–2018]. URL: http://www.aquabalance.ru (дата обращения: 05.10.2018).
6. *Забелин С. Л., Фроловский В. Д.* Разработка и применение метаэвристических алгоритмов для решения задач геометрического покрытия // Научный вестник НГТУ. 2013. № 2. С. 45–52.
7. *Фроловский В. Д.* Приближенные методы решения *NP*- трудных задач в системах автоматизации проектирования: учебное пособие. Новосибирск: Изд-во НГТУ, 2006. 100 с.
8. *Матренин П. В., Гриф М. Г., Секаев В. Г.* Методы стохастической оптимизации: учебное пособие. Новосибирск: Изд-во НГТУ, 2015. 63 с.
9. *Забелин С. Л., Фроловский В. Д.* Разработка и анализ приближенных методов решения оптимизационных задач геометрического покрытия // Информационные технологии в проектировании и производстве. 2011. № 3. С. 54–58.

Статья поступила в редакцию 08.10.2018;
переработанный вариант – 11.10.2018.

Фроловский Владимир Дмитриевич
д.т.н., профессор кафедры автоматизированных систем управления НГТУ (630073, Новосибирск, проспект Карла Маркса, 20), тел. (383) 346-11-00, e-mail: frolovskij@corp.nstu.ru.

Забелин Леонид Юрьевич
к.т.н., доцент кафедры САПР СибГУТИ (630102, Новосибирск, Кирова 86), тел. (383) 269-82-68, e-mail: zabelinlu@sibsutis.ru.

Забелин Сергей Леонидович
ст. преподаватель кафедры САПР СибГУТИ, e-mail: zabelinlu@gmail.com.

Application of bionic models and methods for solving optimization problems of agro technical irrigation systems design

V. Frolovsky, L. Zabelin, S. Zabelin

In this article, approximate metaheuristic algorithms are implemented and investigated for solving the problem of an orthogonal region covering with the prohibitions circles of different radius using bionic principles. The calculated experiments are carried out both on randomly generated objects and on models of real projects. The results obtained during the experiments are analyzed. Conclusions are made about the effectiveness of methods and their use in real-world problems.

Keywords: geometric coatings, optimization problem, bionic methods, placement of irrigation devices.