Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра «Компьютерные технологии»

В. А. Хан

Отчет по лабораторной работе
«Построение управляющих автоматов с помощью генетических алгоритмов»
Вариант №28

г. Санкт-Петербург
2011

Оглавление

Введение

Цель данной лабораторной работы — исследовать применение альтернативных функций приспособленности и их преимущества или недостатки по сравнению с «классической» функцией приспособленности на примере задачи об «умном муравье».

При выполнении работы использовался программный комплекс для изучения методов глобальной оптимизации *GlOpt* [2], разработанный студентами кафедры «Компьютерные технологии» НИУ ИТМО.

1. Постановка задачи

Задача данной лабораторной — исследовать применение альтернативных функций приспособленности и их преимущества или недостатки по сравнению с «классической» функцией приспособленности.

1.1 Постановка задачи об «умном муравье»

Дано поле 32×32 клетки, расположенное на поверхности тора. В 89 клетках находятся яблоки, остальные клетки пусты (рис.1).

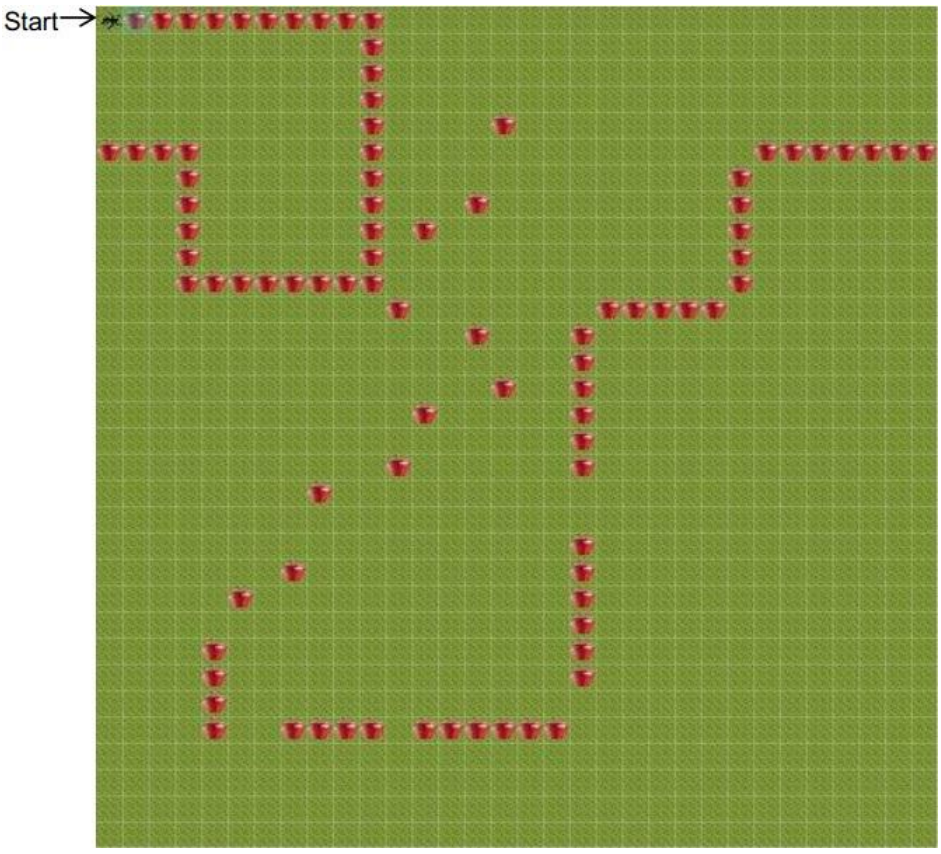

Рис. 1. Поле

Изначально муравей находится в клетке «Start». Муравей видит только клетку перед собой. За ход он может выполнить одно из следующих действий:

- повернуть налево;
- повернуть направо;
- сделать шаг вперед и, если там есть яблоко, съесть его;
- ничего не делать.

Муравей совершает 200 ходов. Требуется построить автомат с определенным числом состояний, управляющий муравьем, такой, что муравей съест наибольшее число яблок за наименьшее число ходов.

1.2 Автомат Мура

Автомат, в котором выходное воздействие зависит только от текущего состояния, называется автоматом Мура.

Автомат Мура представляется совокупностью:

$$A = (S, X, Y, \delta, \mu),$$

где

- S — множество состояний;
- X — множество входных воздействий;
- Y — множество выходных воздействий;
- δ: $S \times Y \rightarrow S$ — таблица переходов;
- μ: $S \rightarrow Y$ — выходные воздействия для каждого состояния.

На рис. 2-4 приведены автоматы, решающие данную задачу.

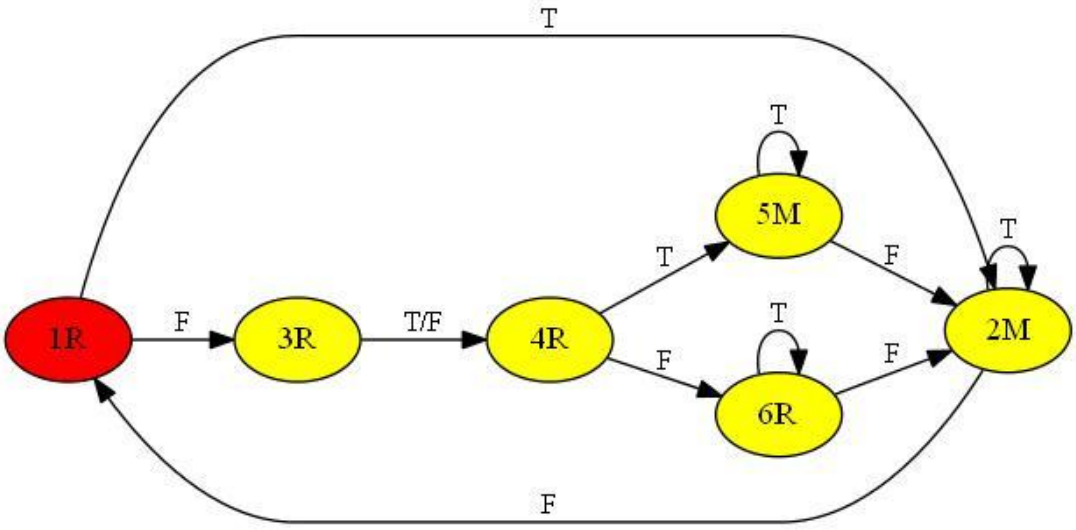

Рис. 2. Автомат Мура с шестью состояниями, съедающий 83 яблока

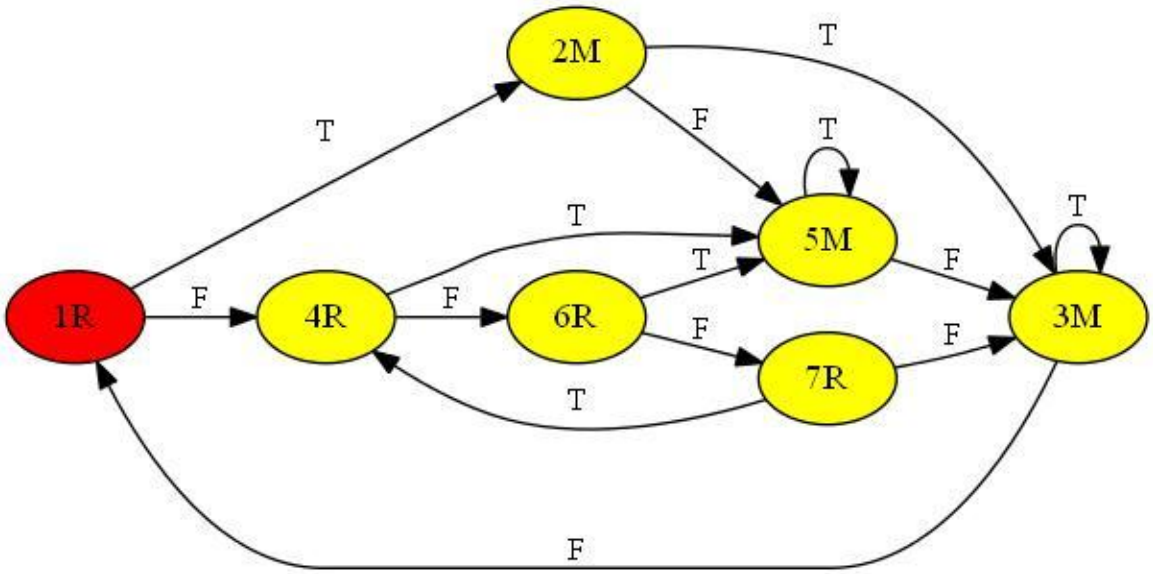

Рис. 3. Автомат Мура с семью состояниями, съедающий 84 яблока

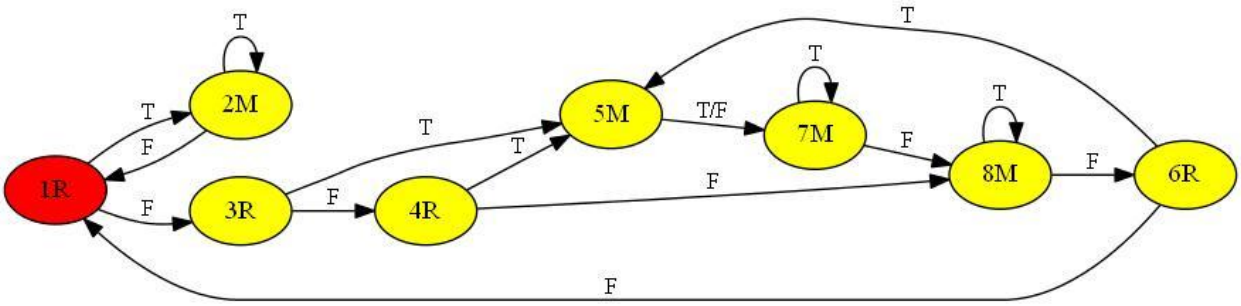

Рис. 4. Автомат Мура с восемью состояниями, съедающий 87 яблок

Стартовое состояние помечено красным цветом. Входные воздействия обозначены следующим образом:

- *T* — перед муравьем есть яблоко;
- *F* — перед муравьем нет яблока.

Действия обозначены так:

- *M* — сделать шаг вперед и съесть яблоко, если оно там есть;
- *L* — повернуть налево;
- *R* — повернуть направо.

Заметим, что во всех приведенных автоматах получили неплохие результаты при отсутствии действия поворота налево.

2. Генетический алгоритм

Для решения задачи поиска эффективного автомата, управляющего муравьем, применяется генетический алгоритм. Работа генетического алгоритма состоит из нескольких фаз. Вначале происходит генерация первого поколения особей, затем алгоритм начинает выполнение итеративного процесса — на каждой итерации строится следующее поколение из предыдущего. При этом применяются следующие процедуры:

- отбор — из предыдущего поколения выбирается часть особей; для сравнения особей между собой алгоритм использует функцию приспособленности, сопоставляющую каждой особи число, определяющее ее приспособленность;
- скрещивание — по двум особям-родителям создаются две новые особи;
- мутация — случайным образом изменяется строение особи.

В этом разделе описывается реализация генетического алгоритма, используемая в данной работе, а также выбранные функции приспособленности.

2.1 Островной генетический алгоритм

В настоящей работе выбрана островная модель генетического алгоритма. Ее особенностью является «расселение» всей популяции на несколько «островов», на каждом из которых эволюция происходит независимо от других. Раз в несколько поколений происходит обмен особями между «островами».

Алгоритм зависит от следующих заданных параметров:

- *ElitePart* — часть лучших особей, переходящих в следующее поколение;
- *MutatedPart* — часть особей, получаемая мутацией;
- *NewPart* — часть особей, создаваемая случайным образом;
- *BigMutationProbability* — вероятность «большой мутации»;
- *MovedPart* — часть особей «острова», перемещаемая на другой «остров» во время «миграции»;
- *MovePeriod* — период «миграций»;
- *IslandPopulation* — «население» каждого «острова»;
- *IslandCount* — количество «островов».

2.2 Метод «рулетки»

Метод «рулетки» выбирает из N особей одну следующим образом: интервал [0,1] разбивается на N отрезков, длины которых пропорциональны

величинам функций приспособленности соответствующих особей. Затем на интервал случайным образом ставится точка. Она попадает на один из *N* отрезков. Выбирается особь, соответствующая этому отрезку.

2.3 Оператор скрещивания

Оператор скрещивания порождает две новые особи из двух особей-родителей. Скрещивание производится следующим образом. Стартовое состояние каждого ребенка полагается равным стартовому состоянию одного из родителей. Затем для каждого состояния случайно выбирается родитель, от которого наследуется действие и таблица переходов.

2.4 Оператор мутации

Оператор мутации изменяет особь одним из следующих равновероятных способов:

- заменяет начальное состояние случайным;
- заменяет действие в случайном состоянии случайным;
- заменяет целевое состояние случайного перехода случайным.

Выбор особей для мутации происходит методом «рулетки».

2.5 Оператор «большой» мутации

Оператор «большой» мутации заселяет выбранные «острова» особями, созданными случайным образом.

Процесс отбора «островов», для которых произойдет «большая» мутация выполняется следующим образом:

1) Для каждого «острова» создается случайное число из промежутка [0,1], которое сравнивается с параметром *BigMutationProbability*.
2) Если полученное число меньше *BigMutationProbability*, то для этого «острова» происходит «большая мутация».

2.6 Генерация очередного поколения

Внутри одного «острова» в следующее поколение попадают:

- часть лучших особей предыдущего поколения (*ElitePart*);
- часть особей, выбранных методом «рулетки», для которых применяется оператор мутации (*MutatedPart*);
- часть особей, созданных случайным образом (*NewPart*);
- оставшаяся часть особей, полученных путем скрещивания.

Генерация очередного поколения происходит следующим образом. Отбираются «острова», для которых произойдет «большая» мутация. Для остальных «островов» генерируется новое поколение вышеуказанным способом.

После обработки всех «островов» определяется необходимость «миграции», которая происходит через определенное количество поколений (*MovePeriod*). Во время «миграции» часть особей с каждого «острова», выбранных методом «рулетки», переселяется на соседний с ним «остров» (*MovedPart*).

2.7 «Классическая» функция приспособленности

«Классическая» функция приспособленности вычисляется по формуле:

$$Fitness = Apples - LastStep / 200,$$

где *Apples* — количество съеденных муравьем яблок за 200 шагов, *LastStep* — номер шага, на котором муравей съел последнее яблоко.

2.8 Альтернативные функции приспособленности

1) Первая альтернативная функция приспособленности ориентирована на скорость съедания яблок муравьем. Функция приспособленности пересчитывается каждые десять действий муравья следующим образом:

$$Fitness = Fitness + Apples / Steps,$$

где *Apples* — количество съеденных муравьем яблок за *Steps* шагов.

2) Вторая альтернативная функция приспособленности ориентирована на количество яблок, съедаемых муравьем на последних действиях. Функция приспособленности пересчитывается каждые десять действий муравья следующим образом:

$$Fitness = Fitness + Apples / (200 - Steps),$$

где *Apples* — количество съеденных муравьем яблок за *Steps* шагов.

3) Третья альтернативная функция приспособленности получена путем объединения «классической» и первой альтернативной функций приспособленности с расчетом на скорость и интенсивность съедания яблок муравьем в начале его пути и общими результатами за 200 шагов. Функция приспособленности вычисляется в два этапа.

На первом этапе функция приспособленности пересчитывается каждые десять действий муравья следующим образом:

$$FirstPart = FirstPart + Apples / Steps,$$

где *Apples* — количество съеденных муравьем яблок за *Steps* шагов.

Итоговая функция вычисляется по формуле:

$$Fitness = FirstPart + AllApples - LastStep / 200,$$

где *AllApples* — количество съеденных муравьем яблок за 200 шагов, *LastStep* — номер шага, на котором муравей съел последнее яблоко.

4) Четвертая альтернативная функция приспособленности получена путем объединения первой и второй альтернативных функций приспособленности с расчетом на скорость и интенсивность съедания яблок муравьем в начале и конце его пути. Функция приспособленности вычисляется в три этапа.

На первом и втором этапе функция приспособленности пересчитываются каждые десять действий муравья следующим образом:

$$FirstPart = FirstPart + Apples / Steps,$$

$$SecondPart = SecondPart + Apples / (200 - Steps),$$

где *Apples* — количество съеденных муравьем яблок за *Steps* шагов.

Итоговая функция вычисляется по формуле:

$$Fitness = FirstPart + SecondPart.$$

5) Пятая альтернативная функция приспособленности ориентирована на скорость съедания яблок муравьем с учетом шага, на котором муравей съел последнее яблоко. Функция приспособленности вычисляется в два этапа.

На первом этапе функция приспособленности пересчитывается каждые десять действий муравья следующим образом:

$$FirstPart = FirstPart + Apples / Steps,$$

где *Apples* — количество съеденных муравьем яблок за *Steps* шагов.

Итоговая функция вычисляется по формуле:

$$Fitness = FirstPart - LastStep / 200,$$

где *LastStep* — номер шага, на котором муравей съел последнее яблоко.

3. Результат работы

Для автоматов, состоящих из $N = 6, 7, 8$ состояний, было проведено 100 запусков с размером поколения 100 особей и числом поколений 1000. В каждом запуске бралось максимальное значение функции приспособленности в соответствующем поколении. После получения результатов ста запусков данные значения усреднялись.

Параметры алгоритма были заданы следующим образом:

- *ElitePart* = 0,1;
- *MutatedPart* = 0,1;
- *NewPart* = 0,1;
- *BigMutationProbability* = 0,001;
- *MovedPart* = 0,1;
- *MovePeriod* = 10;
- *IslandPopulation* = 10;
- *IslandCount* = 10.

Результаты запусков отображены на рис. 5 (для автомата с шестью состояниями), рис. 6 (для автомата с семью состояниями) и рис. 7 (для автомата с восемью состояниями). Для удобного сравнения значений «классической» и альтернативных функций приспособленности величины альтернативных функций на графиках изображены в значениях «классической» функции.

Можно заметить, что наилучшие результаты получены при использовании «классической» и третьей и четвертой альтернативных функций приспособленности. Таким образом, можно сделать вывод, что с помощью хорошо подобранной альтернативной функции приспособленности можно увеличить эффективность работы генетического алгоритма.

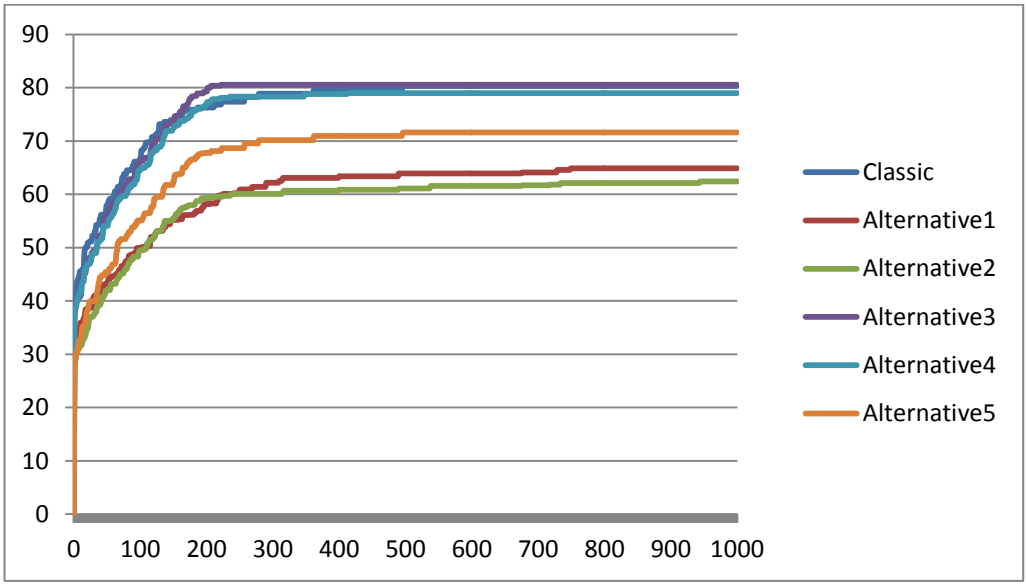

Рис. 5. График зависимости функций приспособленности от количества поколений для автомата с шестью состояниями: 100 запусков, размер популяции 100, число поколений 1000

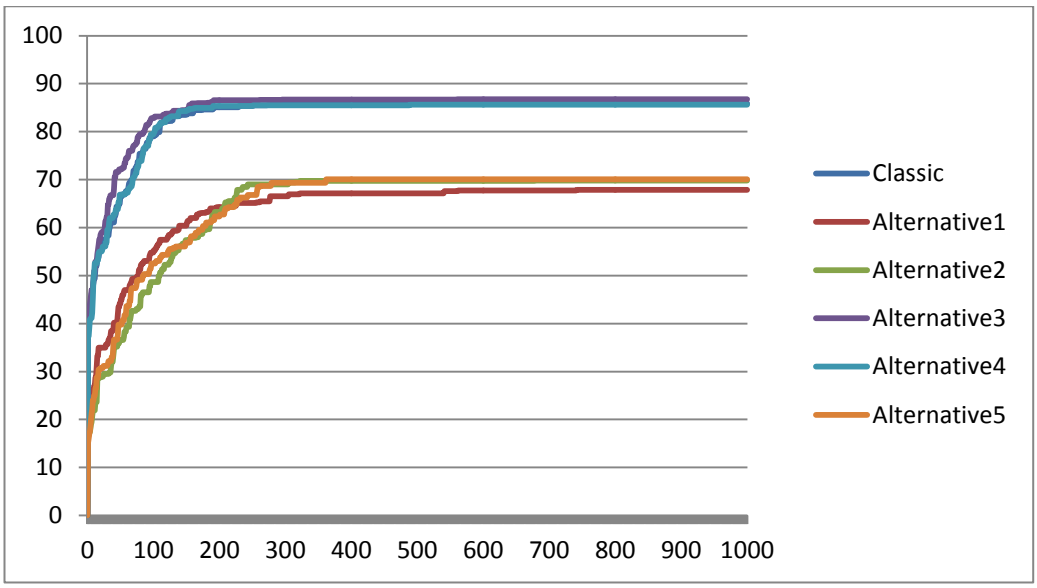

Рис. 6. График зависимости функций приспособленности от количества поколений для автомата с семью состояниями: 100 запусков, размер популяции 100, число поколений 1000

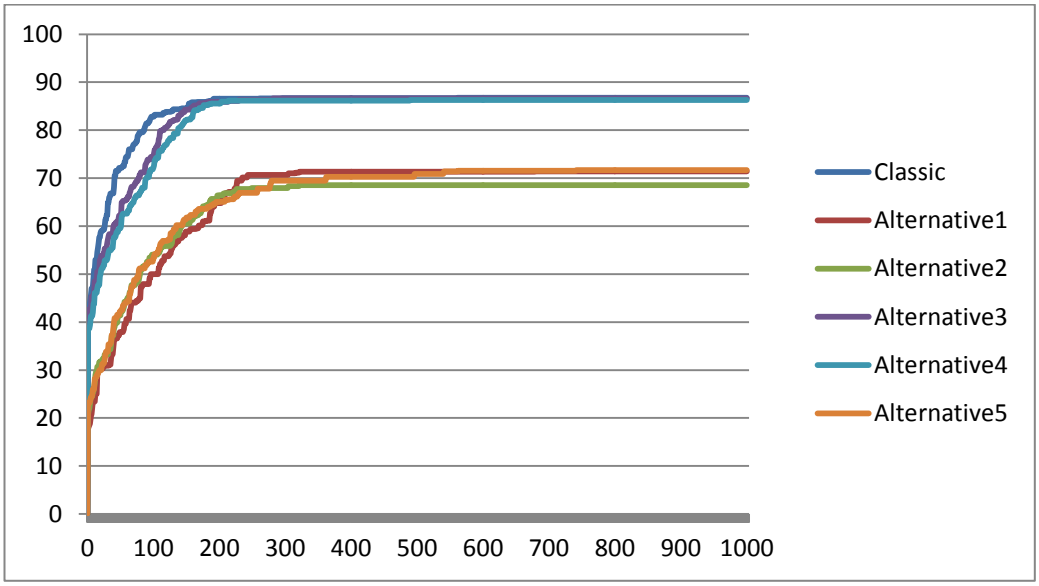

Рис. 7. График зависимости функций приспособленности от количества поколений для автомата с восемью состояниями: 100 запусков, размер популяции 100, число поколений 1000

Заключение

Результаты лабораторной работы показали, что использование хорошо подобранной альтернативной функции приспособленности может увеличить эффективность работы генетического алгоритма при построении автомата Мура для шести, семи и восьми состояний. В качестве такой функции приспособленности для решения задачи об «умном муравье» можно порекомендовать третью альтернативную функцию приспособленности, приведенную в разделе 2.8.

Источники

1. *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. — СПб.: Питер, 2009.

2. *Тяхти А.С., Чебатуркин А.А.* Программный комплекс для изучения методов искусственного интеллекта «Виртуальная лаборатория GlOpt» // НИУ ИТМО, кафедра компьютерных технологий, 2010.
http://is.ifmo.ru/courses/_giopt-src.rar

3. *Яминов Б.* Генетические алгоритмы
http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005

4. *Давыдов А.А., Соколов Д.О., Царев Ф.Н., Шалыто А.А.* Применение островного генетического алгоритма для построения автомата Мура и систем взаимодействующих автоматов Мили на примере задачи об «Умном муравье».
http://is.ifmo.ru/genalg/_scm2008_sokolov.pdf