Труды международных научно-технических конференций «Интеллектуальные системы (IEEE AIS'07)» и «Интеллектуальные САПР (CAD-2007)». – М.: Физматлит, 2007. – С.94-101.

УДК 004.83; 004.89:004.4

ECWORKSHOP – ИНСТРУМЕНТАЛЬНАЯ БИБЛИОТЕКА КЛАССОВ ДЛЯ ЭВОЛЮЦИОННЫХ ВЫЧИСЛЕНИЙ*

Ю.Р. Цой[1]

Представлена разработанная инструментальная библиотека классов для эволюционных вычислений – ECWorkshop. При разработке библиотеки учтены современные требования к программным средствам для эволюционных вычислений, включающие: общность представления структур данных и значений приспособленности, гибкость в использовании эволюционных операторов, а также богатые возможности по управлению параметрами запуска эволюционных алгоритмов и возможности конфигурации файлов отчета. Особенностью реализации является использование технологии паттернов проектирования, что позволило создать модульную библиотеку с широкими возможностями расширения и высокой степенью повторного использования существующих компонент.

Введение

Исследование современных средств вычислительного интеллекта во многом зависит от используемых программных средств, от их быстродействия, гибкости, удобства использования. В эволюционных вычислениях традиционно весьма высока степень повторного использования компонент эволюционного поиска (стратегия селекции, механизмы подстройки параметров эволюционного алгоритма (ЭА), общая схема эволюционного поиска), поэтому становится актуальным вопрос разработки базовых инструментальных средств, позволяющих повторно использовать такие компоненты. Это существенно уменьшает время подготовки эксперимента и анализа полученных экспериментальных данных за счет того, что много-

* Работа выполнена при финансовой поддержке РФФИ (проект № 06-08-00840)

[1] *634034, Томск, пр. Ленина, 30, ТПУ, qai@mail.ru*

кратно используются однажды написанные программные модули.

В данной статье описывается разработанная инструментальная библиотека для эволюционных вычислений ECWorkshop, созданная для облегчения задач исследования и практического применения ЭА.

1. Краткий обзор существующих инструментальных библиотек

В настоящее время наиболее известными и функциональными являются следующие инструментальные библиотеки для эволюционных вычислений:

- Open BEAGLE (Beagle Engine is an Advanced Genetic Learning Environment) [Gagne, Parizeau, 2003].
- ECJ (Evolutionary Computation in Java) [Luke et al., 2005].
- EO (Evolving Objects) [Keijzer et al., 2001].

Библиотеки Open BEAGLE и EO реализована на языке C++, а библиотека ECJ – на языке программирования Java. Все рассматриваемые библиотеки разработаны с использованием объектно-ориентированных принципов и представляют мощные и функциональные средства для решения различных исследовательских и практических задач с использованием эволюционного подхода. В частности, для повышения гибкости в этих библиотеках используются следующие возможности:

- работа с различными способами представления данных;
- работа с различными способами записи приспособленности, не зависящими от способа представления данных и операторов;
- работа с различными операторами;
- работа с различными моделями эволюции;
- управление параметрами запуска ЭА;
- конфигурируемый вывод информации о результатах работы ЭА.

Библиотека ECJ является популярной библиотекой для эволюционных вычислений, написанной с использованием языка программирования Java, но ее недостатком является требовательность к вычислительным ресурсам [Prechelt, 2000] и низкое быстродействие.

Библиотека EO изначально разрабатывалась для работы с любыми структурами данными и содержит богатый набор инструментальных и вспомогательных программных структур и функций [Keijzer et al., 2001]. Имеются возможности гибкой настройки параметров запуска и сбора статистики. Среди недостатков EO отметим сложность работы с этой библиотекой [Gagne, Parizeau, 2003] и отсутствие возможности сбора статистики по многократным запускам эволюционного алгоритма.

В библиотеке Open BEAGLE представлены богатые возможности для программирования эволюционных алгоритмов. Сама библиотека разрабо-

тана с учетом современных подходов и требований к разработке программного обеспечения. Используется модель управления памятью на основе подсчета ссылок и умных указателей, а для управления нестандартными/аварийными ситуациями применяется обработка исключений. Для определения параметров работы ЭА, свойств внутренних структур данных и вывода данных в файл отчета используется формат XML. Достоинством библиотеки Open BEAGLE является обширная и подробная документация по использованию и установке, а также легкость работы с имеющимися инструментальными средствами. Недостатком является сложность внутренней организации библиотеки Open BEAGLE, которая влечет сложность расширения библиотеки. При этом при добавлении новых объектов необходимости учитывать ряд требований, отрицательно сказывающихся на быстродействии (обязательное определение методов `read` и `write`, работающих с форматом XML). Быстродействие также снижается за счет использования в Open BEAGLE механизма обработки исключений [Майерс, 2006].

2. Требования к библиотеке классов

Современные взгляды на разработку программного обеспечения (ПО), и инструментальных библиотек в частности, включают следующие требования:

1. *Модульность*. Необходимо для обеспечения возможности независимого проектирования, разработки и сопровождения (обновления/замены) отдельных элементов (модулей) программы без влияния на работоспособность и стабильность всей программы в целом.

2. Возможность *повторного использования* (*reusability*) отдельных модулей при различных сценариях работы программы. Необходимо для уменьшения времени разработки и размера исполняемого кода, а также для удобства сопровождения библиотеки.

3. *Расширяемость*. Необходимо для дальнейшего роста функциональности библиотеки. Для выполнения этого требования важно выполнение требования *модульности*, так как расширяемость подразумевает также добавление элементов, не предусмотренных на этапе проектирования библиотеки, которое не должно привести к модификации исходных текстов библиотеки классов и/или нарушению стабильности работы.

4. *Мультиплатформенность*. Необходимо для возможности создания программ, использующих библиотеку, независимо от используемой операционной системы или программно-аппаратной платформы.

5. Библиотека классов не должна определять особенности пользовательского интерфейса конечной программы. Необходимо для обеспечения

гибкости использования библиотеки.

Для удовлетворения всех вышеперечисленных требований будем использовать *паттерны проектирования* (*design patterns*) [Гамма и др., 2003], которые разрабатывались для создания гибкого и надежного ПО, предусматривающего повторное использование, и появились в результате обобщения опыта работы многих разработчиков ПО.

Отметим, что помимо «архитектурных» требований, перечисленных выше, существенным является требование к *эффективности* разрабатываемого ПО, что необходимо для обеспечения высокого быстродействия при разумных затратах вычислительных ресурсов.

3. Общее описание разработанной библиотеки

Укрупненная структурная схема разработанной инструментальной библиотеки показана на рис. 1. Основными модулями библиотеки являются:

1) Производящий модуль.
2) Эволюционный алгоритм.
3) Пространство задачи.
4) Модуль обработки результатов.

Рис. 1. Укрупненная схема организации модулей библиотеки

Все входные параметры, определяющие свойства эволюционного алгоритма, задаются в файле конфигурации в формате XML и записываются в структуру `QEAParameters`, содержащую поля доступные для чтения всем модулям и операторам библиотеки.

Производящий модуль реализован с использованием паттерна Фабрика (Factory) [Гамма и др., 2003], и используется для создания всех операторов и структур данных ЭА. Необходимость использования фабричного паттерна вызвана тем, что используемые эволюционные операторы зависят от используемого способа кодирования решений. Такой подход позволяет упростить программный код ЭА за счет абстрагирования от способа кодирования и избежать случая, когда эволюционные операторы не подходят для работы с выбранным способом кодирования.

Сгенерированные производящим модулем структуры данных и операторы используются в модуле, реализующем *эволюционный алгоритм*, для осуществления эволюционного поиска. При этом оценивание приспособленности особей производится путем обращения из модуля эволюционного алгоритма в модуль *пространства задачи* с передачей указателя на оцениваемую особь. В процессе работы модуля «Эволюционный алгоритм» могут быть выполнены множество независимых запусков ЭА. При этом промежуточные и конечные результаты работы ЭА в каждом запуске сохраняются в структуре `QRunSummary`. Отметим, что содержимое структуры `QRunSummary`, содержащей данные о результатах работы текущего запуска ЭА, доступно для чтения всем эволюционным операторам.

Полученные в результате работы модуля «Эволюционный алгоритм» структуры типа `QRunSummary` обрабатываются в *модуле обработки* для проведения статистического анализа результатов и сравнения с результатами работы ЭА с различными настройками параметров.

4. Классы эволюционных операторов и задачи

К эволюционным операторам относятся все классы, используемые в процессе эволюционного поиска для модификации популяции. К таким классам относятся следующие базовые классы:

- `QInitializer` – базовый класс оператора для инициализации популяции.
- `QSelection` – базовый класс оператора для селекции особей.
- `QParentSelector` – базовый класс оператора для выбора особей для скрещивания.
- `QCrossoverOperator` – базовый класс оператора скрещивания.
- `QMutationOperator` – базовый класс оператора мутации.
- `QPopulationSizer` – базовый класс оператора изменения размера

популяции.

Отметим, что возможно расширение приведенного выше списка путем добавления новых операторов.

Использование большинства эволюционных операторов перечисленных выше стандартно, однако работа оператора `QPopulationSizer` требует пояснений. Рассматриваемый оператор используется для изменения размера популяции путем изменения размера популяции потомков (популяция следующего поколения). Поскольку популяция потомков формируется оператором кроссинговера (`QCrossoverOperator`), то для определения точки останова в процессе формирования популяции потомком необходимо использовать оператор `QPopulationSizer`. Простейший вариант оператора `QPopulationSizer` не изменяет размер популяции, оставляя его равным начальному. Такая функциональность реализована в классе `QConstPopulationSizer`.

Пример схемы использования / взаимодействия различных операторов для генетического алгоритма и их связь со структурами данных представлен на рис. 2. Блок «Параметры ЭА; Результаты» содержит данные о параметрах запуска ЭА и текущих результатах работы ЭА.

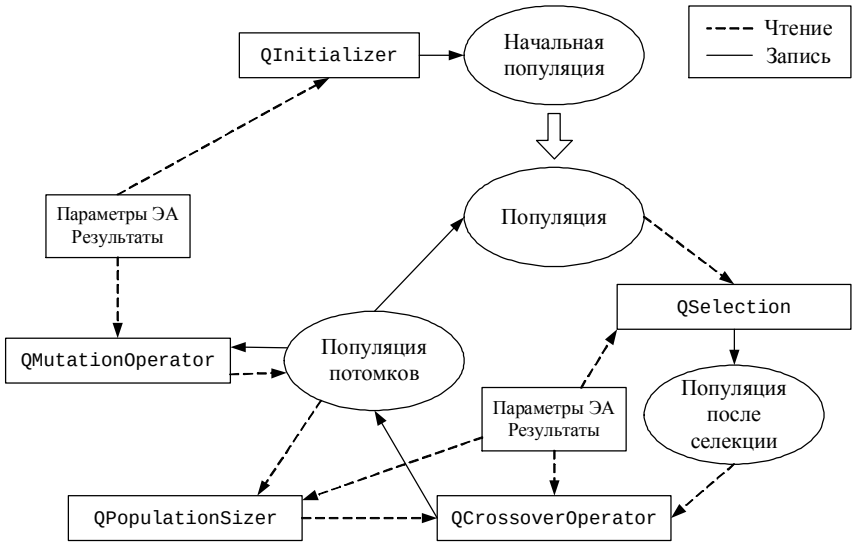

Рис. 2. Пример использования эволюционных операторов для генетического алгоритма. Пунктирными стрелками показаны операции чтения, сплошными – операции записи. Блок «Параметры ЭА; Результаты» указан два раза для упрощения схемы

Для обеспечения требования модульности внутренняя логика работы операторов не зависит от других операторов. Другими словами функционирование одного оператора не влияет на внутреннюю логику функционирования другого оператора (хотя влияние на результат работы возможно).

Для вычисления приспособленности особей используется класс задачи `QProblem`. Использование отдельного класса для вычисления приспособленности позволяет «разделить» эволюционный поиск и особенности решаемой задачи [Tsoy, Spitsyn, 2004]. Тем самым достигается возможность реализации единого набора классов для решения различных оптимизационных задач.

Отметим, что в реализации библиотеки рассматривается только задача *минимизации* приспособленности, поэтому при реализации значение приспособленности особи должно *уменьшаться* с ростом качества соответствующего решения.

Заключение

Описывается разработанная инструментальная библиотека классов для эволюционных вычислений ECWorkshop. Особенностями библиотеки является реализация модульной архитектуры с использованием паттернов проектирования. Благодаря этому достигается гибкость и расширяемость библиотеки. Использование языка программирования C++ позволяет достичь высокого быстродействия входящих в библиотеку модулей.

В настоящее время в библиотеке реализованы следующие способы генетического кодирования информации:

1. *Целочисленное*, для традиционного представления данных в виде бинарных хромосом.

2. *Строками* произвольной арности для кодирования последовательностей символов произвольного алфавита.

3. *Перестановки* для решения комбинаторных задач, в которых значения оптимизируемых параметров не должны повторяться.

4. *Вещественное*, для решения задач оптимизации с вещественными параметрами.

Для каждого из вышеперечисленных способов реализованы базовые операторы скрещивания и мутации. Структура библиотеки позволяет реализовывать операторы общего назначения (селекция, изменение размера популяции и др.) и схемы эволюционного поиска вне зависимости от используемого способа кодирования.

В результате работы ЭА производится сбор и первичная обработка экспериментальных данных по результатам многократных запусков. В част-

ности доступны следующие данные:

1. Изменение по поколениям среднего, лучшего и худшего значений приспособленностей, усредненные по всем запускам, а также значения дисперсии приспособленности в популяции в каждом поколении.

2. Изменение дисперсии величин, перечисленных в п.1, по различным запускам для статистического сравнения результатов работы ЭА с различными настройками параметров.

3. Данные о времени работы ЭА в секундах, а также данные о номере поколения, на котором достигнута сходимость ЭА.

4. Данные о времени поиска решения в секундах и количестве поколений, необходимых для поиска решения.

5. Дополнительные данные об изменении параметров ЭА (например, изменение размера популяции в зависимости от номера поколения).

Исходные коды библиотеки защищены лицензией GPL – General Public Licence для разработок с открытым исходным кодом и доступны для скачивания на странице библиотеки: http://qai.narod.ru/ecw/. На этой же странице можно скачать доступную документацию.

Отметим, что работа над библиотекой не закончена. Планируется добавление новых типов кодирования, операторов и схем эволюционного поиска, а также расширение круга решаемых задач и повышение возможностей конфигурации библиотеки.

Список литературы

[Гамма и др., 2003] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2003.

[Майерс, 2006] Майерс С. Эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов: Пер. с англ. – М.: ДМК Пресс; Спб: Питер, 2006.

[Gagne, Parizeau, 2003] Gagne C., Parizeau M. Genericity in evolutionary computation software tools: principles and case-study // International Journal on Artificial Intelligence Tools. – 2006. – Vol. 15, no. 2. – P. 173–174.

[Keijzer et al., 2001] Keijzer M., Merelo J.J., Romero G., Schoenauer M. Evolving objects: a general purpose evolutionary computation library // In EA-01, Evolution Artificielle, 5th International Conference in Evolutionary Algorithms, 2001.

[Luke et al., 2005] Luke S., Panait L., Balan G., Paus S., Skolicki Z., Bassett J., Hubley R., Chircop A. ECJ13: A java-based evolutionary computation and genetic programming research system // 2005 // http://cs.gmu.edu/?eclab/projects/ecj.

[Prechelt, 2000] Prechelt L. An empirical comparison of seven programming languages // IEEE Computer. – 2000. – Vol. 33, no. 10. – P. 23–29.

[Tsoy, Spitsyn, 2004] Tsoy Y.R., Spitsyn V.G. Use of Design Patterns for Design of the Software Environment for Researches in Genetic Algorithms // Proceedings of 8-th

Korea-Russia International Symposium on Science and Technology KORUS-2004. – Tomsk, 2004. – Pp. 166-168.