© 2012 г. О.Г. МОНАХОВ, канд. техн. наук
(Институт вычислительной математики и математической геофизики СО РАН, Новосибирск)

МАСШТАБИРУЕМЫЙ ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ДЛЯ ОПТИМИЗАЦИИ ФИНАНСОВЫХ СТРАТЕГИЙ НА ГИБРИДНОМ КЛАСТЕРЕ С ГРАФИЧЕСКИМИ ПРОЦЕССОРАМИ [1]

Описан подход для оптимизации торговых стратегий (алгоритмов), основанный на индикаторах финансовых и товарных рынков и эволюционных вычислениях. Представлен параллельный генетический алгоритм, который был реализованный на гибридном кластере с графическими процессорами NVIDIA в технологии CUDA для автоматизации поиска оптимальных параметров торговых стратегий с точки зрения максимизации показателей доходности.

SCALABLE PARALLEL GENETIC ALGORITHMS FOR OPTIMIZATION OF FINANCIAL STRATEGIES ON HYBRID CLUSTER WITH GPU / O.G. Monakhov (Institute of Computational Mathematics and Mathematical Geophysics SB RAS, 6 Pr. Lavrentieva, Novosibirsk, 630090, Russia, E-mail: monakhov@rav.sscc.ru). An approach for optimization of trading strategies (algorithms) based on indicators of financial markets and evolutionary computation is described. Parallel version of a genetic algorithm for search of optimal parameters of trading strategies for maximization of trading profit on hybrid cluster with GPU from NVIDIA is presented.

1. Введение и постановка задачи.

Одним из основных направлений при выработке стратегий биржевой торговли (торговых алгоритмов) является технический анализ ценовых рядов с помощью множества индикаторов[1]-[6]. В соответствии с принятой торговой стратегией, выраженной в виде набора правил, с поведением ценового ряда и значениями индикаторов инвестор принимает решение о совершении/не совершении сделки купли-продажи в данный момент времени. Использование параллельных вычислительных систем для биржевой торговли позволяет компьютерным программам (торговым роботам) самостоятельно отслеживать данные по нескольким индексам на фондовых биржах, оптимизировать торговые стратегии и совершать миллионы сделок за максимально короткий промежуток времени.

Целью данной работы является описание подхода к оптимизации торговых стратегий, основанного на эволюционных вычислениях. Представлен параллельный генетический алгоритм (ГА), реализованный на гибридном кластере с графическими процессорами NVIDIA в технологии CUDA, который в процессе торговых сессий осуществляет

[1]Работа выполнена при финансовой поддержке РФФИ (грант № 08-01-00857).

автоматический поиск оптимальных параметров торговых стратегий и индикаторов с точки зрения максимизации показателей доходности.

Будем считать, что цена на акцию представлена в виде ценового ряда $\{C_i\}$, $1 \leqslant i \leqslant N$, с заданной частотой τ (например, минутные или часовые цены), где C_i - цена закрытия в момент i. Пусть $r_{i+1} = C_{i+1} - C_i$.

Важными инструментами технического анализа рынка акций являются скользящие средние, индикаторы и осцилляторы, на основе которых формируются множество торговых стратегий и которые помогают инвестору принимать решения о купле-продаже акций [1]-[6].

Пусть мы имеем индикатор технического анализа: $I_i^{(n)} = f(C_i, C_{i-1}, ..., C_{i-n})$.

Обобщенная торговая стратегия $S(I_i^{(n)})$, основанная на индикаторе $I_i^{(n)}$, определяется следующими соотношениями:

$$\varphi_{i+1} = \begin{cases} 1, & \text{if } I_i^{(n)} > \varepsilon_1, \\ \varphi_i, & \text{if } -\varepsilon_2 \leqslant I_i^{(n)} \leqslant \varepsilon_1, \\ -1, & \text{if } I_i^{(n)} < -\varepsilon_2, \end{cases}$$

где $\varepsilon_1, \varepsilon_2 > 0$ - уровни значимого изменения индикатора $I_i^{(n)}$.

Состояние покупки в данной торговой стратегии наступает при $\varphi_{i+1} = 1$, а состояние продажи наступает при $\varphi_{i+1} = -1$. Решение о сделке (купли/продаже) принимается при смене состояний: $\varphi_i \varphi_{i+1} = -1$.

Эта стратегия $S(I_i^{(n)})$ будет использована как темплейт (с некоторыми модификациями) для определения торговых стратегий на основе различных индикаторов технического анализа и поиск оптимальных значений свободных параметров (n, ε), определяющих стратегию с наилучшими показателями доходности, будет осуществляться с помощью ГА.

Например, одним из часто используемых индикаторов при анализе ценовых рядов является экспоненциальное скользящее среднее порядка k:

$$\overline{C}_{i+1}^{(k)} = \overline{C}_i^{(k)} + (2/(k+1))(C_{i+1} - \overline{C}_i^{(k)}),$$

$0 \leqslant i \leqslant N-1$, $\overline{C}_0^{(k)} = C_0$. Порядок скользящего среднего k определяет степень сглаживания цены: чем больше k, тем сильнее сглаживание.

Рассчитывается также разность экспоненциальных скользящих средних порядков $k_1 \leqslant k_2$:

$$\overline{r}_i = \frac{\overline{C}_i^{(k_1)} - \overline{C}_i^{(k_2)}}{\overline{C}_i^{(k_2)}}.$$

Приведем пример простейшей торговой стратегии на основе экспоненциальных скользящих средних [2]. Задается уровень значимого изменения сглаженных цен $\varepsilon > 0$. Состояние покупки в данной торговой стратегии наступает при $\overline{r}_i > \varepsilon$, а состояние продажи наступает при $\overline{r}_i < -\varepsilon$. Решение о сделке (купли/продаже) принимается при смене состояний. Стратегия имеет три свободных параметра k_1, k_2, ε, изменение которых изменяет показатели доходности и риска торговой стратегии. Поиск оптимальных стратегий (с наилучшими показателями доходности и/или риска) может осуществлятся

для каждого типа акций отдельно в динамике торговых сессий, с постоянной адаптацией к рыночной ситуации, или в квазидинамическом режиме, когда расчет оптимальных параметров происходит либо через заданные периоды времени либо по выполнению определенных условий (например, по достижении заданного уровня потерь). Частота работы алгоритма оптимизации параметров стратегий при биржевой торговле зависит от динамики рынка и горизонта инвестирования, особенно часто это происходит при работе внутри дня на малых тайм-фреймах (минутных и тиковых интервалах), когда оценка и адаптация параметров стратегий происходит в реальном времени через 100-200 интервалов.

Пусть торговая стратегия S содержит параметры $P = \{p_n\}$, $n \geqslant 0$, описывающие значения целочисленных и действительных коэффициентов и переменных, значения индексов, параметры структур данных, константы и некоторые примитивные операции алгоритма (величины инкрементов и декрементов, знаки переменных, логические операции и отношения, типы округления переменных).

Целевая функция F оценивает величину доходности стратегии S, полученную при заданных значениях параметров $P = \{p_n\}$ и при входных данных ценового ряда C_i : $F_i = F_i(S(P, C_j))$, $j \leqslant i$, $1 \leqslant i \leqslant N$.

Таким образом, проблема оптимизации торговой стратегии состоит в следующем: для данной стратегии S и заданного набора значений ценового ряда C_i , $1 \leqslant i \leqslant N$, необходимо найти такие значения параметров P^* стратегии S, что

$$F_N(S(P^*, C_i)) \geqslant F_N(S(P, C_i))$$

для $1 \leqslant i \leqslant N$, при любых других значениях параметров $P \in Dom(P)$.

Для решения данной проблемы в работе предлагается подход основанный на применении параллельной версии [10] генетических алгоритмов (ГА) [7, 8] с использованием предварительного знания прикладной области (множества индикаторов), выборе обобщенной схемы торговой стратегии, задаваемой в виде темплейта с параметрами [9], и ограничении пространства поиска оптимальных параметров.

2. Генетический алгоритм.

Генетический алгоритм основан на моделировании процесса естественного отбора в популяции особей, каждая из которых представлена точкой в пространстве решений задачи оптимизации. Особи представлены структурами данных Gen - хромосомами, включающими свободные (неопределенные) параметры p_k торговой стратегии S: $Gen = \{P\} = \{p_1, p_2, ..., p_k\}$, $k \geqslant 0$. Эти параметры определяют необходимую торговую стратегию $S(Gen)$. Каждая популяция является множеством структур данных Gen и определяет множество стратегии $S(Gen)$.

Основная идея алгоритма синтеза состоит в эволюционном преобразовании множества хромосом (параметров стратегии) в процессе естественного отбора с целью выживания "сильнейшего". В нашем случае этими особями являются стратегии, имеющие наибольшее значение целевой функции. Алгоритм начинается с генерации начальной популяции. Все особи в этой популяции создаются случайно, затем отбираются наилучшие особи и запоминаются. Для создания популяции следующего поколения (следующей итерации), новые особи формируются с помощью генетических операций селекции (отбора), мутации, кроссовера и добавления новых элементов (для сохранения разнообразия популяции).

Примем, что целевая функция (fitness function, функция качества, функция пригодности) F вычисляет суммарную доходность D_N, полученную в результате торговли в соответствии с данной стратегией S за N шагов для заданного ценового ряда $\{C_i\}$, $1 \leqslant i \leqslant N$:

$$F = D_N = \sum_{m=1}^{N_{br}} (d_m^{br} - Comm),$$

где

$$d_m^{br} = \frac{C_m^{sell} - C_m^{buy}}{C_m^{buy}},$$

C_m^{sell}, C_m^{buy} - цены продажи и покупки в m-той сделке, N_{br} - число сделок за N шагов моделирования, $Comm$ - размер постоянных комиссионных за каждую сделку.

Целью алгоритма является поиск максимума F.

2.1. Представление данных.

Основными структурами данных в программной реализации эволюционного алгоритма являются хромосомы Gen. Для представления и реализации хромосомы была предложена линейная структура для параметров p_k. Линейная структура хромосомы Gen используется для представления различных типов параметров p_k стратегии, таких, как значения целочисленных и действительных коэффициентов и переменных, значения индексов, величины инкрементов и декрементов, а также знаков переменных, логических операций и отношений, типов округления переменных [9, 10].

Приведем пример линейной структуры хромосомы, каждый ген которой обозначен через [g] и соответствует одному из параметров стратегии:

$$\{[\ 5\ ][1.2][-1][+][\&][\geqslant][\lfloor x \rfloor]\}.$$

При создании хромосом Gen задаются значения параметров p_k, по которым можно производить оценивание и модификации стратегии $S(Gen)$.

Таким образом, для фиксированных значений параметров p_k мы можем вычислять значения целевой функции F на основе заданных стратегий $S(Gen)$ и полученных в ходе эволюции хромосом Gen для требуемого ценового ряда $\{C_i\}$, $1 \leqslant i \leqslant N$. После выполнения стратегий S для данного ценового ряда, мы получаем значения целевой функции F и выбираем лучшие стратегии в популяции.

2.2. Операторы алгоритма.

Оператор *мутации* применяется к особям, случайно выбранным из текущей популяции с вероятностью $p_m \in [0, 1]$. Мутация линейной хромосомы Gen состоит в изменении значения случайно выбранного параметра p_k на другую, случайно выбранную величину из множества допустимых значений.

Оператор *кроссовера* (скрещивания) применяется к двум особям (родителям), случайно выбранным из текущей популяции с вероятностью $p_c \in [0, 1]$. Кроссовер состоит в порождении двух новых особей путем обмена частями хромосом родителей (обмен подчастями линейных структур хромосомы).

Оператор создания *нового элемента* (особи) состоит в генерации случайных значений параметров p_k.

Оператор *селекции* (отбора) реализует принцип выживания наиболее приспособленных особей. Он выбирает наилучших особей с минимальными значениями целевой функции.

Заметим, что только простейшие генетические операторы были использованы в алгоритме, но данный подход позволяет применять и более сложные операторы, разработанные для ГА [7, 8].

2.3. Итерационный процесс.

Для поиска оптимума заданной целевой функции F итерационный процесс вычислений в ГА организован следующим образом.

Первая итерация: порождение начальной популяции. Все особи популяции создаются с помощью оператора *новый элемент*, с проверкой и отсеиванием всех непригодных особей. После заполнения массива популяции лучшие особи отбираются и запоминаются в массиве *best*.

Промежуточная итерация: шаг от текущей к следующей популяции. Основной шаг алгоритма состоит в создании нового поколения особей на основе массива *best* и текущей популяции, используя операции селекции, мутации, кроссовера и добавления новых элементов.

После оценки целевой функции для каждой особи в поколении проводится сравнение величин этих функций с величинами целевых функций тех особей, которые сохранены в массиве *best*. В том случае, если элемент из нового поколения лучше, чем элемент $best[i]$, для некоторого i, помещаем новый элемент на место i в массив *best* и сдвигаем в нем все остальные элементы на единицу вниз. Таким образом, лучшие элементы локализуются в верхней части массива *best*.

Последняя итерация (критерий остановки): итерации заканчиваются либо после исполнения заданного числа шагов, либо после нахождения оптимальной стратегии $S(Gen)$ (с заданным значением целевой функции F). После выполнения данного количества шагов алгоритма мы получаем множество (популяцию) стратегий $S(Gen)$, содержащее в элементе $best[0]$ стратегию $S^*(Gen)$, имеющую максимальное значение целевой функции F.

3. Распараллеливание генетического алгоритма и экспериментальные результаты.

Предложенный генетический алгоритм с использованием темплейтов был успешно применен для адаптивной оптимизации торговых стратегий, основанных на следующих, наиболее популярных инструментах технического анализа: экспоненциальных скользящих средних (EMA - exponential moving average), индекса относительной силы (RSI - relative strength index), темпа изменения цены (ROC - price rate-of-change), момента (Momentum) , метода схождения-расхождения скользящих средних (MACD - Moving Average Convergence/Divergence) [1]-[6].

Для эксперимеров были рассмотрены ценовые ряды с минутными интервалами для акций ГАЗПРОМа (тип 1, 10000 точек) и индекса DJIA - Dow Jones Industrial Average (тип 2, 10000 точек), для периода с 16.04.2006 по 16.06.2006. Первые 5000 точек были использованы для обучения, а остальные точки - для тестирования.

Число итераций и размер популяции выбирались экспериментальным путем, основываясь на параметрах из [7, 8]. Значения основных параметров в экспериментах приведены в Табл.1.

Таблица 1. Значения параметров

Параметр	Значение
Размер популяции	(1 ÷ 105)*524288
Число итераций	100
Частота кроссовера	70%
Частота мутации	15%

Генетический алгоритм оптимизации торговых стратегий был реализован в системе эволюционного синтеза алгоритмов на основе шаблонов (TES - template-based evolutionary synthesis) [9] на языке программирования C. Параллельная реализация генетического алгоритма оптимизации стратегий биржевой торговли была выполнена для кластерной суперЭВМ НКС-30Т с гибридной архитектурой (содержащей 40 вычислительных модулей, каждый из которых содержит 2 CPU Intel Xeon X5670 (по 6 ядер) и 3 графических процессора Tesla M 2090 (по 512 ядер), всего 480 ядер CPU и 61440 ядер GPU). Программа реализована в системе программирования CUDA с использованием библиотеки MPI, путем распараллеливания по данным с равномерным распределением популяции по потокам графической подсистемы. На каждом вычислительном модуле использовались 3 ядра CPU (3 потока MPI, по одному на каждую карту, для организации коммуникаций) и все 1536 ядер GPU - для вычислений. В конце итераций среди всех потоков выбирается лучшее решение, что минимизирует взаимодействия и позволяет получить отличное масштабирование и значительное (линейное) ускорение для параллельного алгоритма. Величина популяции составляла 524288 на каждой видеокарте (графическом процессоре), использовались ценовые ряды в 10000 точек. Отметим, что в случае реализации генетического алгоритма на ГПУ, данные для обучения и тестирования следует по возможности помещать или в быструю разделяемую память или в константную память. Так, размещение этих данных в константной памяти, которую могут использовать все потоки сразу, позволило сократить время исполнения алгоритма на ГПУ почти в два раза.

В Табл.2 приведены результаты сравнения для параллельной реализации генетического алгоритма для оптимизации стратегии с MACD для акций ГАЗПРОМа при числе вычислительных модулей M_{CM}, числе графических процессоров N_{GPU}, числе ядер GPU K_{GPU}, размере популяции Pop, времени исполнения T(сек.), и полученном ускорении Sp по отношению к 1 процессору GPU. Из Табл.2 видно, что при линейно возрастающем объёме вычислений (при линейном росте величины популяции), время исполнения остаётся постоянным (с отклонениями не более 1%), что свидетельствует об отличном масштабировании, высокой эффективности распараллеливания и линейном ускорении для параллельного генетического алгоритма .

Используемый генетический алгоритм позволил найти значения параметров торговых стратегий, обеспечивающие увеличение функции суммарной доходности (на 14%

Таблица 2. Сравнение параллельных реализаций генетического алгоритма

M_{CM}	N_{GPU}	K_{GPU}	Pop	T(сек)	Sp
1	1	512	524288	785,45	1
1	3	1536	3*524288	779,4	3
10	30	15360	30*524288	784,17	30
20	60	30720	60*524288	785,42	60
30	90	46080	90*524288	784,47	90

- 155% для различных индикаторов, см. Табл.3) по сравнению с известными ранее [1],[3],[6].

Таблица 3. Увеличение суммарной доходности торговых стратегий

Тип акции \ Индикатор	EMA	$MACD$	RSI	ROC
ГАЗПРОМ	155%	14.3%	75%	67.5%
DJIA	41%	35.5%	31%	16.1%

4. Заключение.

Представленый подход к оптимизации торговых стратегий, основанный на индикаторах технического анализа, эволюционных вычислениях и темплейтах, был успешно применен для поиска свободных параметров стратегий с целью максимизации функции суммарной доходности.

Параллельная реализация генетического алгоритма оптимизации стратегий биржевой торговли на на гибридном кластере с графическими процессорами NVIDIA позволяет получить отличное масштабирование и существенное (линейное) ускорение по отношению к 1 процессору GPU, а также увеличить значения функции суммарной доходности.

СПИСОК ЛИТЕРАТУРЫ

1. *Achelis, S.B.* Technical analysis from A to Z. Chicago: Probus. 1996.

2. *Артемьев С. С., Якунин М. А.* Математическое и статистическое моделирование на фондовых рынках. - Новосибирск: ИВМиМГ СО РАН, 2003.

3. *LeBeau, Charles and Lucas, David W.* Computer analysis of the futures market, New-York: IRWIN, 1992.

4. *Weissman, Richard L.* Mechanical Trading Systems, Hoboken, New Jersey: John Wiley and Sons, Inc., 2005.

5. *Salov, Valerii*, Modeling maximum trading profits with C++ : new trading and money management concepts, Hoboken, New Jersey: John Wiley and Sons, Inc., 2007.

6. *Blau, W.* Momentum, Direction and Divergence, Hoboken, New Jersey: John Wiley and Sons, Inc., 2001.

7. *Goldberg, D. E.* Genetic Algorithms, in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989.

8. *Koza, J.* Genetic Programming. Cambridge: The MIT Press,1992.

9. *Монахов О. Г.* Эволюционный синтез алгоритмов на основе шаблонов // Автометрия. 2006. N1. С.106-116.

10. *Монахов О.Г., Монахова Э.А.* Параллельные системы с распределенной памятью: структуры и организация взаимодействий, Новосибирск: Изд-во СО РАН, 2000.

11. *Боресков А.В., Харламов А.А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2010.