ПРОГРАММИРОВАНИЕ

УДК 519.6

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ АРХИТЕКТУРЫ CUDA ДЛЯ АППРОКСИМАЦИИ МНОЖЕСТВА ПАРЕТО С ПОМОЩЬЮ МЕТОДА РОЯ ЧАСТИЦ

А.Э. Антух, А.П. Карпенко, А.С. Семенихин

EFFICIENCY RESEARCH OF CUDA APPLICATION FOR PARETO SET RECONSTRUCTION USING PARTICLE SWARM OPTIMIZATION

A.E. Antukh, A.P. Karpenko, A.S. Semenikhin

Во многих практически значимых случаях при решении задачи многокритериальной оптимизации предварительно целесообразно построить аппроксимацию множества Парето этой задачи. Рассматривается комбинация известного метода приближенного построения множества Парето «недоминируемая сортировка» и метода глобальной оптимизации роем частиц. Целью работы является исследование эффективности указанной комбинации методов при их реализации на графических процессорных устройствах с архитектурой CUDA.

Ключевые слова: метод роя частиц, ГПУ, множество Парето.

There are a lot of practical cases that requires Pareto set construction to solve multi-objective optimization task. This article is devoted to observe a combination of well-known non-dominated sorting method with particle swarm optimization algorithm. Authors set a goal to research efficiency of the new combined method on graphical computational unit with CUDA architecture.

Keywords: particle swarm optimization, GPU, Pareto set.

Введение

Использование графических ускорителей (ГПУ) для научных расчетов началось относительно недавно [1]. Во многих прикладных областях до сих пор остается открытым вопрос об эффективности ГПУ по сравнению с классическими компьютерными системами. Целью данной работы является исследование эффективности ГПУ с архитектурой CUDA для задачи аппроксимации множества Парето с помощью метода роя частиц (PSO).

Основная масса публикаций, посвященных CUDA-вычислениям, ориентирована на задачи, обладающие явным параллелизмом по данным [2]. Такие задачи хорошо распараллеливаются и позволяют получать существенный прирост производительности вычислений по сравнению с последовательными вычислениями на хост-ЭВМ. В задачах многокритериальной оптимизации не всегда присутствует параллелизм по данным, и эффективность решения таких задач на ГПУ с архитектурой CUDA мало освещена в литературе. Исследование эффективности CUDA-вычислений при решении задачи однокритериальной оптимизации выполнено, например, в работе [3].

В параграфе 1 работы приведена математическая формулировка задачи многокритериальной оптимизации и дано определение множества Парето. Параграф 2 содержит краткое описание используемого алгоритма метода роя частиц. В параграфе 3 приведено описание особенностей реализации алгоритма на ГПУ, а также дана постановка тестовой задачи многокритериальной оптимизации. Параграф 4 содержит результаты экспериментов и оценку эффективности алгоритма. В заключении подведены итоги работы и определены направления ее развития.

1. Постановка задачи

Задана совокупность частных критериев оптимальности $f_1(X), f_2(X), \dots, f_m(X)$, которые образуют векторный критерий $F(X)$. Здесь X – n-мерный вектор варьируемых параметров [4]. Ставится задача минимизации каждого из указанных критериев в одной и той же области допустимых значений вектора варьируемых параметров $D_X \in \Pi \cap D$, где $\Pi = \{X | x_i^- \le x_i \le x_i^+, i \in [1:n]\}$ – «технологический» параллелепипед, $D = \{X | g_1(X) \ge 0, g_2(X) \ge 0, \dots\}$. Здесь $g_1(X), g_2(X), \dots$ – ограничивающие функции. Условно задача многокритериальной оптимизации записывается в виде

$$\min_{X \in D_X} F(X) = F(X^\bullet). \tag{1}$$

Векторный критерий оптимальности $F(X)$ выполняет отображение множества D_X в некоторое множество D_F пространства критериев, которое называется множеством достижимости задачи (1). Введем на множестве D_F отношение предпочтения. Будем говорить, что вектор $F_1 = F(X_1) \in D_F$ предпочтительнее вектора $F_2 = F(X_2) \in D_F$ (или вектор F_1 доминирует вектор F_2), и писать $F_1 \succ F_2$, если среди неравенств $f_k(X_1) \le f_k(X_2), k \in [1:m]$ имеется хотя бы одно строгое. Выделим из множества D_F подмножество точек $D_F^\bullet$ (фронт Парето), для которых нет более предпочтительных. Множество $D_X^\bullet \in D_X$, соответствующее множеству $D_F^\bullet$, называется множеством Парето [4]. Таким образом, если $X \in D_X^\bullet$, то $F(X) \in D_F^\bullet$.

Если $F(X_1) \succ F(X_2)$, то будем говорить, что вектор X_1 предпочтительнее вектора X_2 (или вектор X_1 доминирует вектор X_2), и писать $X_1 \succ X_2$.

Ставится задача приближенного построения множества Парето $D_X^\bullet$ или, что то же самое, фронта Парето $D_F^\bullet$, задачи многокритериальной оптимизации (1).

2. Методы роя частиц и недоминируемой сортировки

Множество частиц обозначим $P = \{P_i, i \in [1:N]\}$, где N – число частиц в рое (размер популяции). В дискретный момент времени $t \in [0:T]$ координаты частицы P_i определяются вектором $X_{i,t} = (x_{i,t,1}, x_{i,t,2}, \dots, x_{i,t,n})$, а ее скорость – вектором $V_{i,t} = (v_{i,t,1}, v_{i,t,2}, \dots, v_{i,t,n})$; T – число итераций. Начальные координаты и скорости частицы P_i определяют векторы $X_{i,0} = X_i^0$, $V_{i,0} = V_i^0$ соответственно.

Итерации в каноническом методе PSO выполняются по схеме [5]

$$V_{i,t+1} = \alpha V_{i,t} + U_1[0,\beta] \otimes (X_{i,t}^b - X_{i,t}) + U_2[0,\gamma](X_{g,t} - X_{i,t}), \tag{2}$$

$$X_{i,t+1} = X_{i,t} + V_{i,t+1}. \tag{3}$$

Здесь $U[a,b]$ представляет собой n-мерный вектор псевдослучайных чисел, равномерно распределенных в интервале $[a,b]$; $\otimes$ – символ покомпонентного умножения векторов; $X_{i,t}^b$ – вектор координат частицы P_i с наилучшим значением целевой функции $F(X)$ за все время поиска $[0:t]$; $X_{g,t}$ – вектор координат соседней с данной частицы с наилучшим за

то же время значением целевой функции $F(X)$; α, β, γ – свободные параметры алгоритма. Важнейшее в методе PSO понятие соседства частиц связано с используемой топологией соседства, и определено,например, в работе [5]. В процессе итераций вектор $X^b_{i,t}$ образует так называемый собственный путь (private guide) частицы P_i, а вектор $X_{g,t}$ – локальный путь (local guide) этой частицы.

Канонический метод роя частиц ориентирован на решение задач однокритериальной оптимизации. Для задачи многокритериальной оптимизации (1) используем модификацию этого метода MOPSO (Multi Objective Particle Swarm Optimization) [6].

Важной частью метода MOPSO является определение глобально-лучшей (в смысле формулы (1)) частицы для каждой частицы в популяции. В силу специфики задачи многокритериальной оптимизации, глобально-лучшая частица отыскивается на множестве Парето.

Псевдокод метода MOPSO представлен на рис. 1. На каждой итерации в процессе выполнения шага 3 метода происходит обновление архива частиц A_t. Функция $Update$ выполняет сравнение частиц текущего поколения с недоминируемыми частицами из архива A_t и определяет те частицы, которые необходимо добавить в архив, а также те частицы, которые следует в архиве заменить. Функция реализует метод недоминируемой сортировки, предложенный в работе [7].

```
Шаг 1: t=0
Шаг 2: Инициализация популяции P_t и архива частиц A_t
       For i=1 to N
           X_{i,0} := X^0_i;  V_{i,0} := V^0_i;  X^b_{i,0} := X^0_i
       End;
           A_t := {};
Шаг 3: A_{t+1} := Update(P_t, A_t);
Шаг 4:
       For i=1 to N
           X_{g,t} := FindGlobalBest(A_{t+1}, X_{i,t});
           For j=1 to n
               v_{i,t+1,j} := α v_{i,t,j} + U_1[0,β] ⊗ (x^b_{i,t,j} − x_{i,t,j}) + U_2[0,γ] ⊗ (x_{g,t,j} − x_{i,t,j});
               x_{i,t+1,j} := x_{i,t,j} + v_{i,t+1,j};
           End;
           If (X_{i,t+1} ≻ X^b_{i,t}) X^b_{i,t+1} := X_{i,t+1} Else X^b_{i,t+1} := X^b_{i,t}
       End;
Шаг 5: If (Критерий останова = false)
       t := t + 1
       GOTO Шаг 3
End.
```

Рис. 1. Псевдокод метода MOPSO

Выбор глобально лучшей частицы осуществляет функция $FindGlobalBest$. Существует несколько способов реализации этой функции. В данной работе используется метод «меняющихся соседей» Хью и Эберхарта [6]. Рассмотрим суть этого метода на примере задачи двухкритериальной оптимизации. Поиск глобально лучшей частицы для каждой частицы популяции осуществляется в пространстве критериев следующим образом. Сначала вычисляем расстояние от частицы P_i до других частиц, содержащихся в архиве A_t, используя значения первого («фиксированного») критерия оптимальности $f_1(X)$. Таким образом, для

частицы P_i находим k ее ближайших локальных соседей. Затем, используя второй критерий $f_2(X)$, из числа указанных k соседей находим наилучшую частицу для частицы P_i, которая и полагается глобально лучшей для этой частицы P_i.

Итерации продолжаются до тех пор, пока множество недоминируемых решений не перестанет меняться, либо до достижения заданного числа итераций.

3. Реализация алгоритма и тестовая задача

Была рассмотрена следующая двухмерная двухкритериальная тестовая задача, точные фронт и множество Парето для которой известны:

$$\begin{cases} f_1(X) = x_1^2 + x_2^2, \\ f_2(X) = (x_1 - 1)^2 + (x_2 - 1)^2; \end{cases} \tag{4}$$

$$D_X = \{X| - 5 \leq x_i \leq 5.i = 1, 2\}. \tag{5}$$

Результаты исследования плотности покрытия множества Парето и фронта Парето задачи (4), (5) при использовании метода MOPSO приведены в публикации [8]. В данной работе исследуется эффективность указанного метода при его реализации на ГПУ.

Исследование выполнено с использованием в качестве хост-ЭВМ персональной ЭВМ, управляемой операционной системой Windows XP, в следующей конфигурации: процессор – Intel Pentium Dual E2160 (1,8 ГГц); оперативная память – 2 ГБ. При вычислениях с использованием только хост-процессора было использовано одно ядро. В качестве ГПУ использована плата NVidia GeForce 8500GT, содержащая 16 потоковых процессоров. Архитектура CUDA рассмотрена, например, в работе [9].

В практически значимых задачах многокритериальной оптимизации основной объем вычислений связан с вычислениями значений частных критериев оптимальности. Например, если речь идет о задаче оптимизации сложной динамической системы, каждое такое вычисление требует численного интегрирования соответствующей системы обыкновенных дифференциальных уравнений. Поэтому в работе принята следующая схема организации вычислений: метод MOPSO реализует хост-процессор системы, а вычисления значений частных критериев оптимальности – ГПУ.

Эффективность параллельных вычислений оценивается с помощью ускорения

$$S = \frac{T_{CPU}}{T_{GPU}}, \tag{6}$$

где T_{CPU} – время решения задачи с использованием только хост-процессора системы, T_{GPU} – аналогичное время при решении задачи с использованием ГПУ. Подчеркнем, что под ускорением S в данном случае понимается повышение производительности вычислений при использовании ГПУ относительно производительности вычислений, производимых только на хост-процессоре.

Важной составляющей частью работы является исследование зависимости ускорения (6) от суммарной вычислительной сложности частных критериев оптимальности. Для моделирования разных вычислительных сложностей критериев (4) использовано их многократное вычисление. В качестве меры вычислительной сложности критериев использован коэффициент сложности C, равный числу их вычислений.

4. Результаты экспериментов

Исследование выполнено при варьировании следующих параметров:

- число итераций $T = 30; 50;$
- коэффициент сложности $C = 1; 10; 100; 1000;$
- число частиц в популяции $N = 32; 64; 128; 256.$

Поскольку эффективность метода существенно зависит от начальных параметров частиц X_i^0, V_i^0,$i \in [1 : N]$ результаты исследования усреднены по этим начальным параметрам.

Некоторые результаты исследования представлены в таблице. Таблица показывает, что, как и следовало ожидать, при невысокой вычислительной сложности частных критериев оптимальности распараллеливание вычислений неэффективно (поскольку в этом случае велика доля коммуникационных расходов). При увеличении вычислительной сложности критериев, а также числа частиц в популяции, ускорение вычислений достигает величины 23.

Таблицу иллюстрируют рисунки 2 и 3.

Таблица

N	**T**	**C**	$\mathbf{T_{CPU}}$**,с**	$\mathbf{T_{GPU}}$**,с**	**S**
32	30	1	0,03	0,16	0,17
		10	0,09	0,18	0,50
		100	0,53	0,41	1,31
		1000	6,92	2,47	2,81
	50	1	0,06	0,21	0,30
		10	0,30	0,27	1,12
		100	2,39	0.66	3,64
		1000	25,64	4,34	5,90
64	30	1	0,06	0,27	0,23
		10	0,30	0,30	1,00
		100	2,52	0,70	3,58
		1000	23,33	4,70	4,96
	50	1	0,15	0,36	0,41
		10	0,64	0,42	1,53
		100	6,56	1,14	5,75
		1000	69,83	7,64	9,14
128	30	1	0,13	0,42	0,31
		10	0,86	0,48	1,77
		100	7,31	1,44	5,09
		1000	79,64	9,25	8,61
	50	1	0,54	0,66	0,82
		10	2,50	0,80	3,14
		100	28,50	2,61	10,93
		1000	219,03	15,41	14,22
128	30	1	1,00	0,74	1,36
		10	2,70	0,97	2,79
		100	18,25	2,53	7,21
		1000	268,36	22,81	11,76
	50	1	3,36	1,23	2,72
		10	12,44	1,50	8,29
		100	81,44	4,45	18,30
		1000	690,44	29,38	23,50

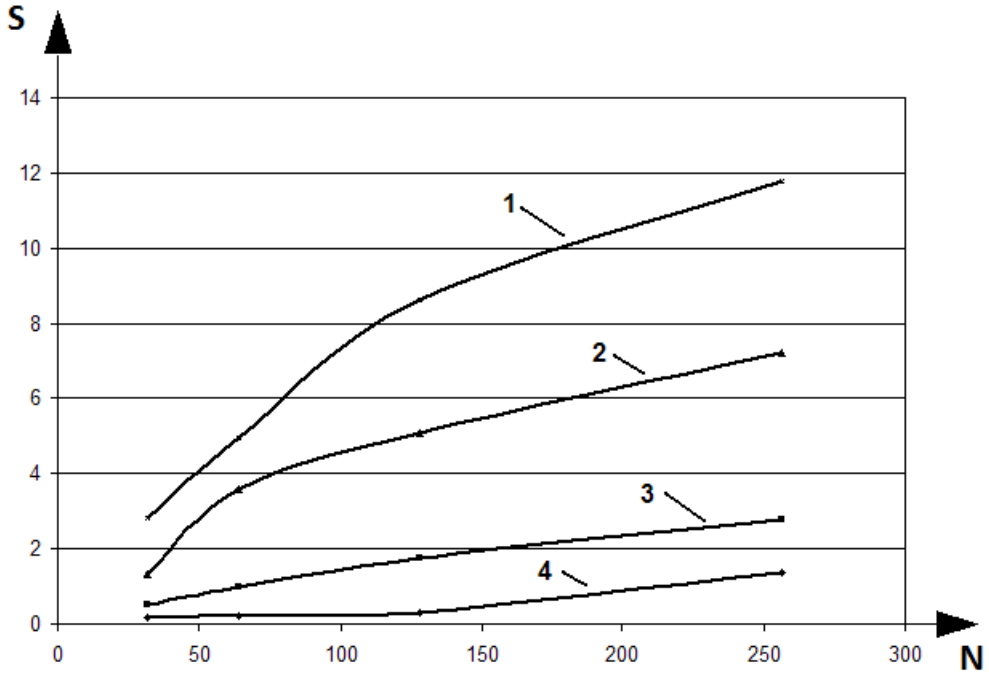

Рис. 2. Ускорение вычислений при числе итераций $T = 30$:

$1 - C = 1000$; $2 - C = 100$; $3 - C = 10$; $4 - C = 1$

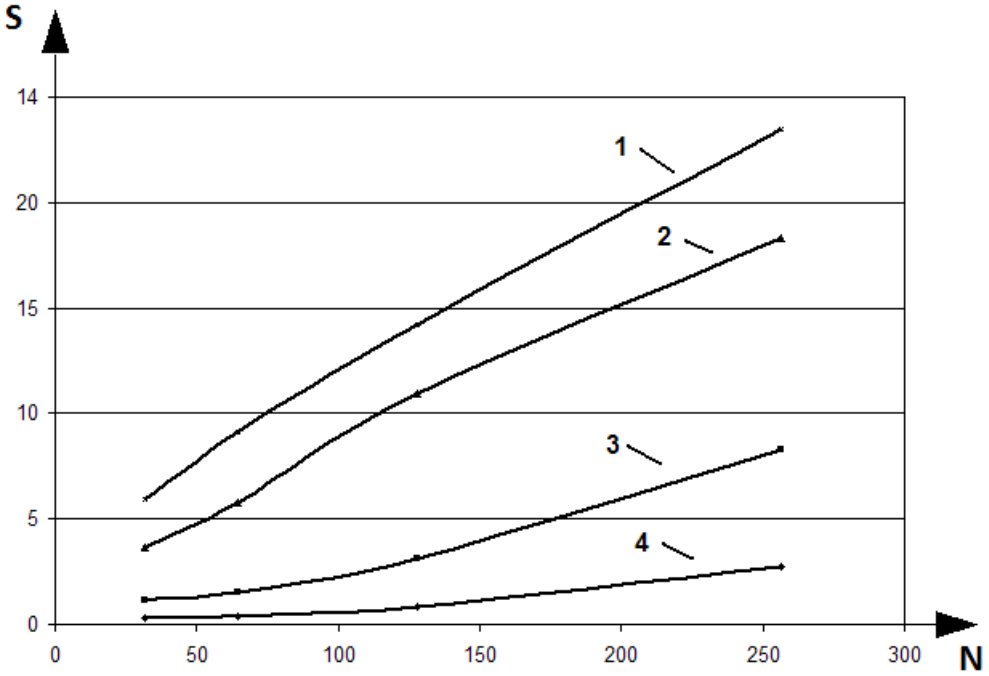

Рис. 3. Ускорение вычислений при числе итераций $T = 50$:

$1 - C = 1000$; $2 - C = 100$; $3 - C = 10$; $4 - C = 1$

Рисунки 2, 3 показывают практически линейный рост ускорения с ростом числа частиц N. Это обстоятельство позволяет надеяться на получение ускорения, превышающего

максимально достигнутое в рассматриваемых экспериментах (равное 23) при дальнейшем увеличении числа частиц в популяции.

5. Заключение

Выполнено исследование эффективности комбинации метода MOPSO и метода недоминируемой сортировки при приближенном построении множества Парето в задаче многокритериальной оптимизации с помощью ГПУ с архитектурой CUDA. Исследование показало высокий потенциал использования графических ускорителей и библиотек для работы с ними (CUDA) для эффективного решения указанной задачи.

В развитие работы предполагается расширение класса тестовых задач многокритериальной оптимизации; определение для ГПУ различных архитектур оптимальных значений свободных параметров рассматриваемых методов, обеспечивающих максимальное ускорение вычислений; теоретическая оценка ускорения при заданных значениях свободных параметров этих методов и архитектуре ГПУ.

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2011».

Литература

1. Вычисления на ГПУ [Электронный ресурс] //
URL: http://www.nvidia.ru/page/gpu_computing.html (дата обращения: 13.12.2010).

2. NVIDIA CUDA C SDK Code [Электронный ресурс] //
URL: http://developer.download.nvidia.com/compute/cuda/sdk/website/samples.html (дата обращения: 13.12.2010).

3. Карпенко, А.П. Глобальная безусловная оптимизация роем частиц на графических процессорах архитектуры CUDA / А.П. Карпенко, Е.Ю. Селиверстов // Наука и образование: электронное научно-техническое издание [Электронный ресурс]. – 2010. – №4. //
URL:http://technomag.edu.ru/doc/142202.html (дата обращения: 13.12.2010).

4. Штойер, Р. Многокритериальная оптимизация. Теория, вычисления и приложения / Р. Штойер. – М.: Радио и связь, 1992. – 504 с.

5. Карпенко, А.П. Глобальная оптимизация методом роя частиц. Обзор / А.П. Карпенко, Е.Ю. Селиверстов // Информационные технологии. – 2010. – № 2. – С. 25 – 34.

6. Hu, X. Multiobjective optimization using dynamic neighborhood particle swarm optimization / X. Hu, R. Eberhart // World Congress on Computational Inelligence. – 2002. – P. 1677 – 1681.

7. Srinivas, N. Multiobjective optimization using nondominated sorting in genetic algorithms / N. Shinvas, K. Deb // Evolutionary Computation. – 1994. – V.2. – P. 221 – 248.

8. Антух, А.Э. Построение множества Парето методом роя частиц на графических процессорах архитектуры CUDA / А.Э. Антух, А.С. Семенихин, Р.В. Хасанова // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: тр. междунар. суперкомпьютер. конф. (21 – 26 сентября 2009 г., г. Новороссийск). – М., 2010. – С. 274 – 280.

9. Фролов, В. Введение в технологию CUDA //
URL: http://cgm.computergraphics.ru/issues/issue16/cuda (дата обращения: 13.12.2010).

References

1. *Vychisleniya na GPU* [GPU calculations] Available at: http://www.nvidia.ru/page/gpu_computing.html (accessed 13.12.2010).
2. *NVIDIA CUDA C SDK Code* Available at: http://developer.download.nvidia.com/compute/cuda/sdk/website/samples.html (accessed 13.12.2010).
3. Karpenko A.P, Seliverstov E.U. Global non-conditional swarm optimizaion using CUDA [Global'naja bezuslovnaja optimizacija roem chastic na graficheskih processorah arhitektury CUDA]. *Nauka i obrazovanie: jelektronnoe nauchno-tehnicheskoe izdanie*, 2010, no. 4, available at: http://technomag.edu.ru/doc/142202.html
4. Shtojer, R. Mnogokriterial'naja optimizacija. Teorija, vychislenija i prilozhenija [*Multimodal optimization. Theory, computing and application*]. Moscow, 1992. 504 p.
5. Karpenko A.P., Seliverstov E.U. Global optimization using partical swarm method. Overview [Global'naja optimizacija metodom roja chastic. Obzor] *Informacionnye tehnologii*, 2010, no. 2, pp. 25 – 34.
6. Hu X., Eberhart R. Multiobjective optimization using dynamic neighborhood particle swarm optimization. World Congress on Computational Inelligence. 2002, pp. 1677 – 1681.
7. Srinivas N., Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation, 1994, vol.2, pp. 221 – 248.
8. Antuh A.E., Semenihin A.S., Hasanova R.V. Pareto set construction usgin PSO on GPU with CUDA library. *Nauchnyj servis v seti Internet: superkomp'juternye centry i zadachi: tr. mezhdunar. superkomp'juter. konf. (21 – 26 sentjabrja 2009 g., g. Novorossijsk)*. Moscow, 2010, pp. 274 – 280.
9. Frolov V. *Vvedenie v tehnologiju CUDA* [CUDA Introduction]. Available at: http://cgm.computergraphics.ru/issues/issue16/cuda (accessed 13.12.2010).

Анатолий Павлович Карпенко, доктор физико-математических наук, профессор, кафедра «Системы автоматизированного проектирования», МГТУ имени Н.Э. Баумана (Россия, г. Москва), apkarpenko@mail.ru.

Anatoliy Pavlovich Karpenko, Doctor of Physico-mathematical Sciences, Full Professor, CAD/CAM/CAE department, Bauman Moscow State Technical University (Russia, Moscow), apkarpenko@mail.ru.

Артем Сергеевич Семенихин, аспирант, кафедра «Системы автоматизированного проектирования», МГТУ имени Н.Э. Баумана (Россия, г. Москва), saspost@yandex.ru.

Artyom Sergeevich Semenikhin, Postgraduate Student, CAD/CAM/CAE department, Bauman Moscow State Technical University (Russia, Moscow), saspost@yandex.ru.

Александр Эдуардович Антух, студент, кафедра «Системы автоматизированного проектирования», МГТУ имени Н.Э. Баумана (Россия, г. Москва), alexander.antukh@gmail.com.

Alexander Eduardovich Antukh, student, CAD/CAM/CAE department, Bauman Moscow State Technical University (Russia, Moscow), alexander.antukh@gmail.com.

Поступила в редакцию 4 июля 2011 г.