Н. В. Кононова, **И. Н. Рубежная**, **Р. А. Кочкаров**
(Ставрополь, СИЭУ ФПГТУ, СКИБУПК, Москва, ФГОУВПО «Финансовая академия при Правительстве РФ»). **Применение технологии параллельного программирования в решении графовых задач.**

В работе, представленной данным докладом, предложен параллельный алгоритм вершинной раскраски предфрактального графа, смежность старых ребер которого сохраняется, а порождающая затравка — f-хроматический граф. Параллельный алгоритм α построен для PRAM — модели параллельной вычислительной системы.

Для правильной работы алгоритма α важна следующая теорема.

Теорема 1. *Предфрактальный граф $G_L = (V_L, E_L)$, порожденный f-хроматической затравкой $H = (W, Q)$ с сохранением смежности старых ребер, также является f-хроматическим, т. е. $\chi(G_L) = \chi(H) = k$.*

Алгоритм α. Рассмотрим предфрактальный граф $G_L = (V_L, E_L)$, смежность старых ребер которого сохраняется. Пусть G_L порожден f-хроматической затравкой $H = (W, Q)$, у которой $|W| = n$, $|Q| = q$. Алгоритм использует k процессоров $p_1, p_2, \ldots, p_k$, где $k = n^{L-1}$.

Опишем принцип работы алгоритма α. Поскольку порождающая затравка $H = (W, Q)$ является f-хроматической, то для ее правильной и одновременно минимальной раскраски понадобится ровно f цветов, т. е. хроматическое число $\chi(H) = f$, $f \leqslant n$. Отметим, что для раскраски затравки H будет использоваться известный алгоритм минимальной раскраски. Алгоритм минимальной раскраски представим в виде процедуры МинРаскраска, которая вызывается основным алгоритмом в нужное время.

В начале алгоритма процессор p_1 назначается подграф-затравке первого ранга $z_1^{(1)}$. Для раскраски подграф-затравки $z_1^{(1)}$ понадобится f цветов, в соответствии с предположением о f-хроматической затравке H.

Далее для каждой подграф-затравки второго ранга $z_{s_2}^{(2)}$ назначим процессоры p_{s_2}, $s_2 = 1, 2, \ldots, n$. У каждой из них одна вершина оказалась окрашенной, так как подграф-затравка первого ранга имеет одну общую вершину с подграф-затравкой второго ранга, в силу того, что смежность старых ребер сохраняется.

Рассмотрим подграф-затравку $z_1^{(2)}$, одна из ее вершин оказалась окрашенной, например, в цвет 1. Далее, используя процедуру МинРаскраска, окрасим оставшиеся вершины в $(k-1)$ неиспользованных цветов. Окрасим таким же образом все подграф-затравки второго ранга $z_{s_2}^{(2)}$.

Далее рассматриваем подграф-затравки третьего ранга $z_{s_3}^{(3)}$, у каждой из них оказалось окрашенной по одной вершине. Окрашиваем подграф-затравки третьего ранга $z_{s_3}^{(3)}$ с помощью процедуры МинРаскраска в соответствии с описанным раннее.

Дойдя до L-го ранга, назначим каждой подграф-затравке $z_{s_L}^{(L)}$ процессоры p_{s_L}, $s_L = 1, 2, \ldots, n^{L-1}$. Таким образом, окрашивая подграф-затравки до L-го ранга включительно, получим правильную f-раскраску предфрактального графа G_L.

Алгоритм α включает в себя процедуру МинРаскраска. Процедура МинРаскраска используется для минимальной f-раскраски подграф-затравок разных рангов. На вход процедуры подается подграф-затравка с одной окрашенной вершиной, далее оставшиеся вершины подграф-затравки окрашиваются в $(f-1)$ свободных цветов в соответствии с последовательным алгоритмом минимальной раскраски. На выходе процедуры получаем правильную f-раскраску подграф-затравки.

Раскраска графа G_3 фактически ведет к поэтапной раскраске графов G_1 и G_2. Раскраска подграф-затравки $z_1^{(1)}$ предфрактального графа G_L совпадает с раскраской графа G_1, а раскраска подграф-затравки $z_1^{(1)}$ и подграф-затравок $z_1^{(2)}, z_2^{(2)}, \ldots, z_4^{(2)}$ с раскраской графа G_3.

Теорема 2. *Вычислительная сложность алгоритма α на предфрактальном*

графе $G_L = (V_L, E_L)$, *порожденного* f*-хроматической затравкой* $H = (W, Q)$ *с сохранением смежности старых ребер, равна* $O(n^2 N)$.

Теорема 3. *Временная сложность алгоритма* α *на предфрактальном графе* $G_L = (V_L, E_L)$, *порожденного* f*-хроматической затравкой* $H = (W, Q)$ *с сохранением смежности старых ребер, равна* $O(n^2 L)$.

Теорема 4. *Ускорение параллельного алгоритма* α *при использовании* k *процессоров,* $k = n^{L-1}$, *от последовательного алгоритма* α *равно* $O(n^L/L)$.