Программная реализация параллельных генетических алгоритмов оптимизации для САПР ТП

Г. В. Верхова[1], С. В. Акимов[2]
Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича» (СПбГУТ)
[1]galina500@inbox.ru, [2]akimov-sv@yandex.ru

***Аннотация*. Представлены результаты программной реализации параллельных генетических алгоритмов для систем автоматизированного проектирования технологических процессов. Рассмотрены особенности распараллеливания генетических алгоритмов и структуры программного обеспечения на языке C# в рамках платформы .NET.**

Ключевые слова: параллельные генетические алгоритмы; САПР ТП; станки с ЧПУ; C#; .NET

I. Введение

Современные тенденции развития промышленности заключаются в постоянном повышении степени автоматизации технологических процессов и производств и интеграции производственных и бизнес процессов на всех этапах жизненного цикла продукции [1–2]. В рамках комплексной автоматизации технологических процессов и производств важную роль играет создание высокоэффективного программного обеспечения для станков с числовым программным управлением (ЧПУ), играющих одну из основных ролей в современном гибком производстве [3]. Станки с числовым программным управлением постоянно эволюционируют, появляются новые виды таких станков, изменяется удельных вес реализуемыми такими станками технологий.

В современном производстве набирают все большую популярность аддитивные технологии [4], а также технологии лазерно-лучевой обработки [5–6]. Данные технологии требуют разработки сложного программно-алгоритмического обеспечения, необходимого для полного раскрытия потенциала этих технологий.

В данной статье представлены результаты разработки и программной реализации параллельных генетических алгоритмов для оптимизации траектории движения исполнительного механизма электронно-лучевой обработки материалов, используемого в станке с числовым программным управлением. Такой вид обработки позволяет создавать гибкие производственные системы, которые могут использоваться как в крупносерийном, так и в мелкосерийном производстве. Он не требует создания шаблонов, а процесс обработки полностью задается программным способом.

Наиболее широкое распространение данного класса обработки материалов получила лазерная обработка. Разработка программно-алгоритмического обеспечения, рассматриваемого в статье, осуществлялась в первую очередь для нее. Однако, рассмотренные алгоритмы могут быть использованы в любых системах электронно-лучевой обработки материалов, а также при любых другой способах обработки плоскостных заготовок.

Для решения задачи оптимизации траектории движения исполнительного механизма были выбраны генетические алгоритмы, так как они хорошо подходят для решения подобного класса задач, допуская возможность эффективного распараллеливания [7]. Возможность распараллеливания алгоритмов является актуальной ввиду широкого распространения многоядерных и многопроцессорных вычислительных систем, а также большой вычислительной сложности решения задачи оптимизации траектории движения исполнительного механизма.

II. Формализация задачи

Схема формализации задачи оптимизации траектории движения исполнительного механизма лазерной обработки станка с числовым программным управлением представлена на рис. 1. Все дорожки, определяющие суммарную траекторию движения, подвергаются сегментации и нумеруются. Каждый сегмент $S_{(i)}$ может быть представлен следующим образом:

$$S_{(i)} = <x_a, y_a, x_b, y_b, l, v, t, d, i>, \quad (1)$$

где x_a, x_b, y_a, y_b – координаты начала и конца сегмента (точки a и b сегмента на рис. 1), соответственно, l – длина сегмента, v – скорость прохождения сегмента, t – время, затрачиваемое на прохождение сегмента, направление прохождения сегмента, i – номер сегмента в очереди обработки.

Список сегментов полностью определяет траекторию обработки:

$$\text{List} = \langle S_{(i,j)} \rangle,\ i, j \in [1, n], \qquad (2)$$

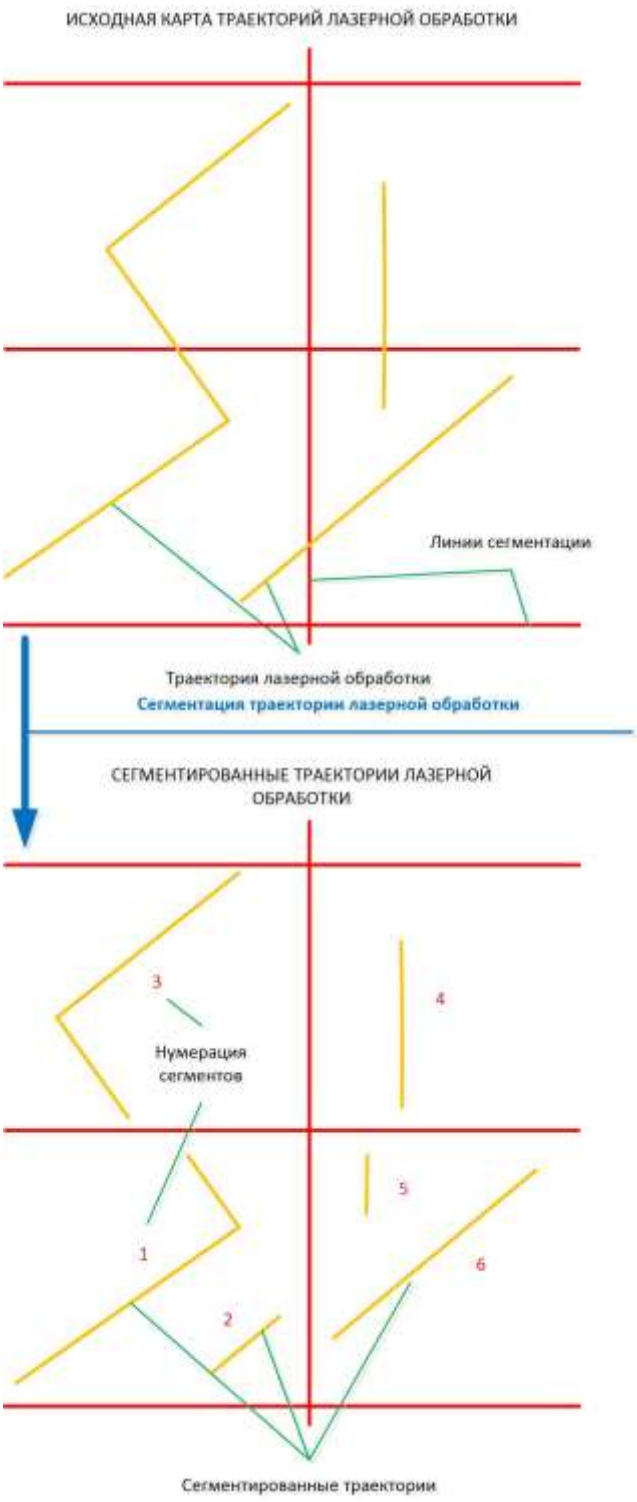

Рис. 1. Формализация процесса оптимизации траектории лазерной обработки

В выражении (2) n – число сегментов, i – номер (идентификатор) сегмента, j – номер сегмента в списке, определяющий очередь обработки. Учитывая (1) и (2) получим время, затрачиваемое на обработку:

$$T = T_t + T_{idle} \qquad (3)$$

Здесь T_t – время, затрачиваемое на обработку, T_{idle} – общее время, затрачиваемое на прохождение кареткой от одного сегмента к другому, плюс время, необходимое для подхода к первому сегменту, и от первого к базе. Отсюда задачу оптимизации можно сформулировать, как нахождение последовательности обработки сегментов и их ориентацию (начало и конец обработки), при которой время T, будет минимальным. Так как время, затрачиваемое на лазерную обработку сегментов не зависит от последовательности обработки сегментов, то интерес представляет минимизация только времени T_{idle}.

III. Базовый генетический алгоритм оптимизации траектории движения исполнительного механизма станка с ЧПУ

Задача определения траектории движения лазерной головки станка с числовым программным управлением может быть решена с применением генетических алгоритмов. Схема формирования хромосомы представлена на рис. 2.

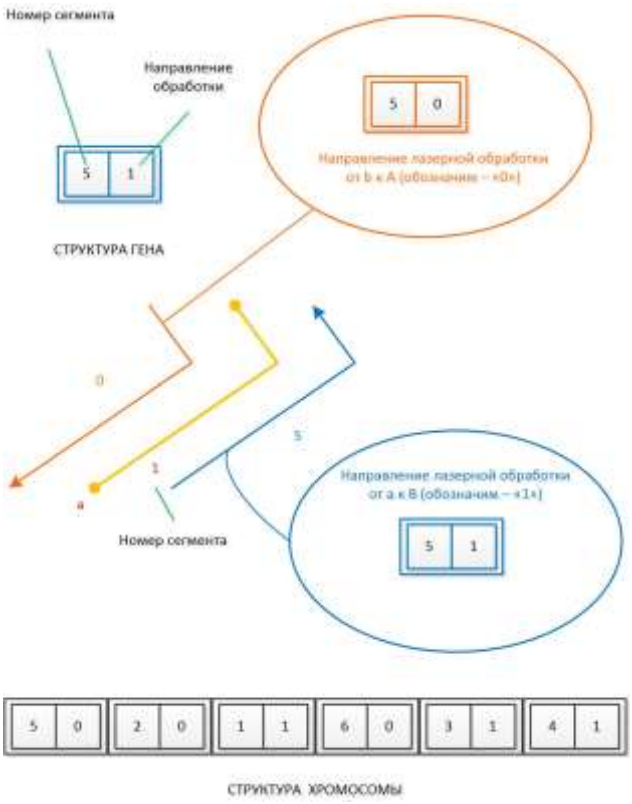

Рис. 2. Структура хромосомы

Каждый отдельный ген содержит информацию о номере сегмента и направлении обработки, который определяет движение лазерной головки вдоль сегмента. Случай, когда лазерная головка движется по направлению от a к b, обозначим как «0», случай движения в противоположном направлении – «1». Последовательность генов в хромосоме определяет последовательность лазерной обработки отдельных сегментов, и таким образом, хромосома однозначно кодирует траекторию движения лазерной головки в процессе обработки изделия.

Учитывая особенность задачи необходимо использовать кроссинговер, сохраняющий все гены, входящие в состав родительских хромосом. Если не будет выполняться данное условие, часть сегментов, подлежащих обработке, не будет обработана, а некоторые сегменты будут обработаны дважды. Данный вид кроссинговеров хорошо изучен, и применяется при решении задач коммивояжера с помощью генетических алгоритмов.

Простейший вид оператора кроссинговера представлен на рис. 3. Другим эффективным видом кроссинговера для данной задачи является циклический кроссинговер, хорошо себя зарекомендовавший при решении задач коммивояжера.

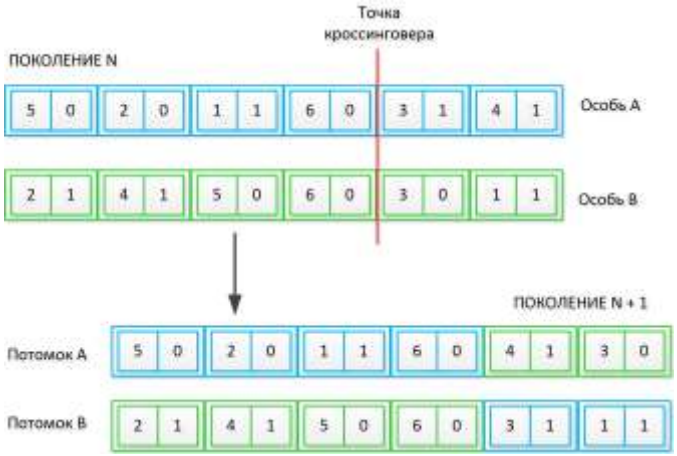

Рис. 3. Оператор кроссинговера

Оператор мутации может заключаться в изменении направления обработки сегмента и в перестановке отдельных сегментов (рис. 4). Возможны и другие виды операторов мутации, значительно сильнее меняющие исходную хромосому.

Исходя из условия задачи видно, что кодирование и оператор кроссинговера во многом похожи на применяемые в задаче коммивояжера [8–9], решаемой с помощью генетических алгоритмов. Отличие заключается в необходимости учета направленности прохождения сегмента *d*. Так как генетические алгоритмы показали свою эффективность при решении задачи коммивояжёра, то имеются все основания полагать, что они окажутся эффективными и для решения задачи оптимизации траектории движения обработке излучением.

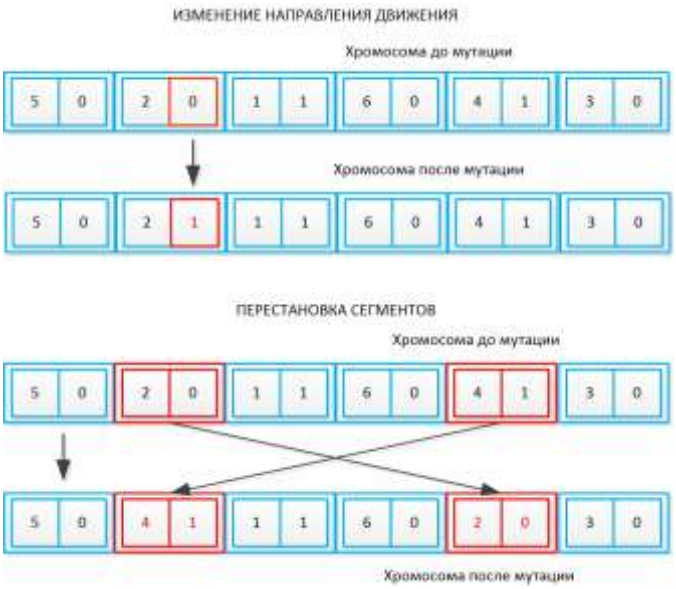

Рис. 4. Операторы мутации

IV. Программная реализация островного алгоритма

Существуют различные модели распараллеливания генетических алгоритмов [10]. Простейшей моделью распараллеливания является модель «рабочий – хозяин», когда вычисления фитнес функции осуществляется параллельно, а сам алгоритм является последовательным. Данный вид алгоритмов подходит для тех задач, где вычисление фитнес функции требует значительных вычислительных затрат. Такая ситуация возникает при использовании сложных моделей оптимизируемых (синтезируемых) объектов, например радиоэлектронных устройств. Однако в нашем случае данная модель не подходит, так как вычисление фитнес функции не требует значительных вычислительных затрат. Также не подходит и клеточная модель.

Учитывая специфику задачи оптимизации траектории движения исполнительного механизма станка с числовым программным управлением, наилучшим кандидатом является островная модель распараллеливания генетических алгоритмов. Схема программной реализации островной модели представлена на рис. 5. Обмен особями между популяциями происходит по циклической схеме.

Данная модель реализуется следующим образом. Из главного потока создаются потоки, реализующие эволюции в островных популяциях. Число островных популяций целесообразно взять равным числу ядер в вычислительной системе, что обеспечит максимальную производительность. Максимальную часть времени в потоках эволюция осуществляется совершенно независимо, что положительно сказывается на эффективности параллельного алгоритма. К популяциям доступ имеет исключительно поток, который ее создал. Это гарантирует отсутствие конфликтов и снижения производительности из-за блокировок, так как никакой другой поток не может попытаться обратиться к данной

популяции. В каждом потоке для моделирования эволюции можно использовать обычные непотокобезопасные вычислительные конструкции.

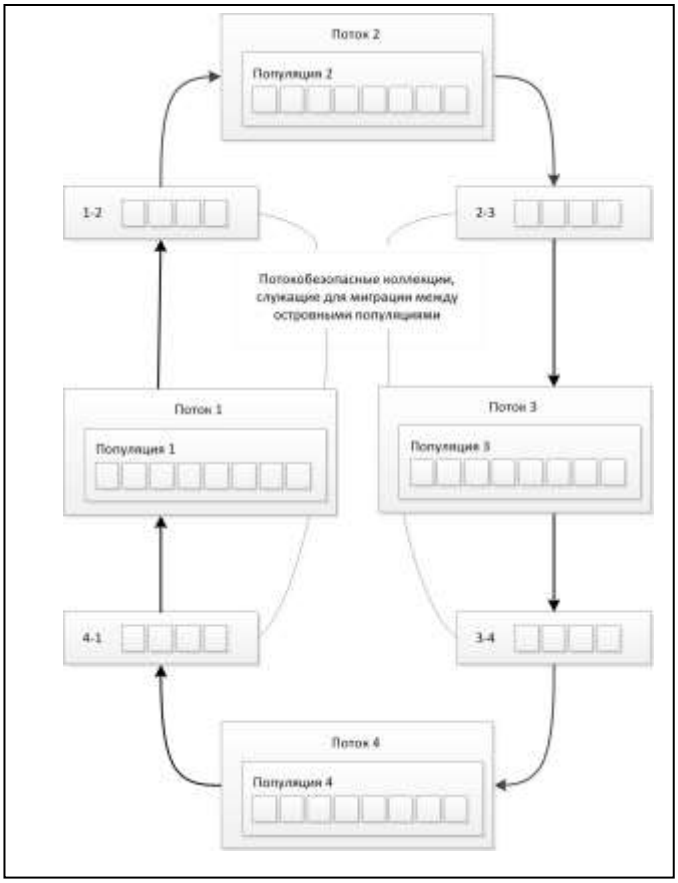

Рис. 5. Схема реализации островной модели

Обмен особями между популяциями осуществляется с помощью потокобезопасных коллекций. У каждого потока имеется доступ к входной и выходной коллекциям. Так, в случае четырехядерного процессора для потока 1, это будут коллекции 1-2 (выходная) и 4-1 (выходная).

Перед началом обработки нового поколения каждый поток проверяет наличие особей во входной коллекции. Если таковые отсутствуют и не сработал механизм начала инициации обмена особями, то запускается обработка нового поколения в эволюционном процессе. Если поток обнаружил наличие особей-мигрантов во входной коллекции, он передает столько же своих особей в выходную коллекцию, и замещает отданные, полученными из входной коллекции.

В случае срабатывания механизма, инициирующего начало миграции, поток передает часть своих особей в выходную коллекцию, и тут же запускает процесс обработки нового поколения на уменьшенной популяции. Тем самым потоки, реализующие эволюции на островных популяциях, никогда не переходит в режим ожидания. Восполнение популяции произойдет после завершения цикла обмена особями всех популяций.

V. Заключение

Реализация данного алгоритма продемонстрировала его высокую эффективность распараллеливания. Последовательно были запрограммированы три варианта алгоритмов: 1) последовательный, 2) распараллеленный с помощью штатных средств платформы .NET и 3) реализующий островная модель. В таблице 1 приведены результаты загрузки четырехядерного процессора.

ТАБЛИЦА I Сравнение загрузки процессора

Вид распараллеливания	*Загрузка процессора, %*	
	Минимальная	*Максимальная*
Последовательный генетический алгоритм	24	25
Расспараллеливания с помощью штатных средств платформы .NET (Parallel.For, Parallel.Foreach)	53	59
Островная модель	94	98

В дальнейшем предполагается продолжить исследования в данной области, исследовав различные стратегии селекции, кроссинговера и мутации, а также адаптации генетического алгоритма.

Список литературы

[1] Samaranayake P., Ramanathan, K., Laosirihongthong T. Implementing industry 4.0 – A technological readiness perspective // 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). 2017, Pp. 529-533.

[2] Verkhova G.V., Akimov S.V. Multi-aspect modeling system objects in CALS // Proceedings of 2017 XX IEEE International conference on soft computing and measurements (SCM) 2017. С. 449-451.

[3] Васильев В. Ю. Тренды развития технологий и производства компонентов и узлов микросистемной техники // Нано- и микросистемная техника. 2016. Т. 18, № 7. С. 403-415.

[4] Nguyen D. S., Vignat F. Topology optimization as an innovative design method for additive manufacturing // 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2017. P. 304-308.

[5] Afanasyev M. Y., Fedosov Y. V., Nemkova A. A. Designing features of power optical units for technological equipment // Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2015, Vol. 16, N 2 (102), P. 244-250. DOI:10.17586/2226-1494-2016-16-2-244-250.

[6] Афанасьев М. Я., Грибовский А. А. Концепция адаптивной платформы технологического оборудования // Изв. вузов. Приборостроение. 2015. Т. 58, № 4. С. 268-272. DOI 10.17586/0021-3454-2015-58-4-268-272.

[7] Baesler F., Palma C. Multiobjective parallel machine scheduling in the sawmill industry using genetic algorithms. The International Journal of Advanced Manufacturing Technology, 74(5). P. 757-768, 2014.

[8] Oliver I. M., Smith D. J., Holland J. R. C. A study of permutation crossover operators on the traveling salesman problem // Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, pages 224–230, MIT, Cambridge, MA, 28-31. July 1987. Lawrence Erlbaum Associates: Hillsdale, New Jersey.

[9] Davoian K. Gorlatch S.. A modified genetic algorithm for the traveling salesman problem and its parallelization // Artificial Intelligence and Applications AIA2005, pages 453–116, Innsbruck (Austria), 14.-16. February 2005.

[10] Емельянов В. В., Курейчик В. В., Курейчик В. М. Теория и практика эволюционного моделирования. М.: Физматлит, 2003. 432 С.