и структур. В результате экспериментов было установлено, что такими значениями являются $P_к = 0,35$; $P_м = 0,1$; M = 80; K = 120.

Вероятность получения оптимального решения после одного прогона генетического алгоритма составила 0,8. Исследования трудоемкости алгоритма показали, что при фиксированных значениях $P_к$ и $P_м$, M и K она имеет оценку O(L). Время выполнения 120 генераций для разнесения 200 цепей на коммутационном поле размером 10 x 10 составило 48 секунд.

Экспериментальные исследования показали, что наилучшие по качеству решения получались при использовании подхода, связанного с распараллеливанием генетического алгоритма. В процессе генетического поиска осуществляется эволюционирование нескольких подпопуляций. На каждой генерации хромосомы случайным образом мигрируют из одной подпопуляции в другую. Исследования показали, что достаточно трех подпопуляций, при этом вероятность оптимального решения составила 0,94. Сравнение с алгоритмами, приведенными [1,2,3] показало, что при том же и даже меньшем времени работы предложенный алгоритм дает более качественные решения. Преимущество особенно заметно для задач большой размерности.

ЛИТЕРАТУРА

1. Naveed Sherwani. Algorithms for VLSI physical design automation // Kluwer academic pablishers. Boston / Dordrecht / London. - 1995.
2. *K.W.Lee and C.Sechen*. A new global ronter for row - based layout // Proceedings of IEEE International Conference on Computer - Aided Design, November, 1998.
3. *C.Chiang, M.Sarrafraden, C.K.Wong*. A Weighted - Steiner - Tree - Based Global Router / Manuscript. - 1992.
4. *Лебедев Б.К.* Метод оптимального распределения ресурсов платы. // Техническая кибернетика. - 1980. - Вып.1. - С. 217.
5. *Лебедев Б.К.* Распределение ресурсов коммутационного поля. // Автоматизация проектирования электронной аппаратуры. - Таганрог: Изд-во ТРТИ, 1988. - Вып.1. - С. 9-92.

УДК 681.3.001.63+007.52:611.81

В.А. Костенко, Р.Л. Смелянский, А.Г. Трекин

СИНТЕЗ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, ОПТИМАЛЬНЫХ ПО ЧИСЛУ ПРОЦЕССОРОВ, С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Введение. Задача синтеза архитектур вычислительных систем (ВС) в самом общем виде может быть поставлена следующим образом. Для заданного поведения прикладной программы *H(PR)* требуется синтезировать архитектуру *HW*, параллельную прикладную программу *HP* и выбрать способ организации параллельного вычислительного процесса φ. При этом должны оптимизироваться критерии оценки качества решения $\{f_i\}$ и выполняться ограничения на допустимые решения $\{g_i\}$. В качестве параметров оптимизации (управляемых переменных) выступают варьируемые параметры моделей *HW*, *HP* и φ. Критерии оценки качества решения, ограничения на допустимые решения и варьируемые параметры моделей определяются при конкретизации задачи синтеза архитектур.

Задача синтеза архитектур ВС относится к классу задач структурного синтеза. В отличии от задач параметрического синтеза, задачи структурного синтеза в общем случае не могут быть отнесены к классу формально разрешимых. При реше-

нии задач структурного синтеза приходится иметь дело с начально не определенными структурными связями, числом и типом компонентов, не метрическими характеристиками компонентов, нечеткими и противоречивыми критериями оценки качества полученного решения для системы в целом и т.п. Данные особенности задач делают их формально неразрешимыми в общем случае.

Формальные модели описания функционирования ВС, предложенные в [1], позволяют определить структуру ВС и программы как конструктивные объекты, композируемые/декомпозируемые из элементарных неделимых компонентов. Задача синтеза архитектур ВС при использовании разработанного в [1] формализма может быть сформулирована как дискретная экстремальная многопараметрическая и многокритериальная задача с ограничениями:

$$f_i(HP,HW,\varphi) \rightarrow min/max,\ i \in I_1,I_2;$$
$$g_i(HP,HW,\varphi) \le 0,\ i \in I_3,I_4;$$
$$g_i(HP,HW,\varphi) = 0,\ i \in I_5,I_6;$$
$$HP \in HP^*,\ HW \in HW^*;\ \varphi \in \varphi^*;$$

где HP - модель параллельной программы, HW - модель архитектуры ВС, φ - оператор интерпретации (соответствует модели организации параллельного вычислительного процесса, т.е. задает взаимодействие между компонентами из HP,HW), f_i - критерии оценки качества решения, g_i - ограничения на допустимые решения, I_1, I_3, I_5 - множества динамических критериев и ограничений, I_2, I_4, I_6 - множества статических критериев и ограничений, (HP^*,HW^*,φ^*) - пространство решений.

Данная задача обладает следующими основными особенностями [2]:

1. Пространство возможных решений (HP^*,HW^*,φ^*) по типу объектов является трехкомпонентным и независимая оптимизация параметров отдельных компонентов невозможна.
2. Значения динамических критериев и ограничений для конкретного варианта решения задачи могут быть определены лишь после выполнения оператора интерпретации φ.
3. f_i, g_i, φ - операторы, заданные правилами/алгоритмами их вычисления, т.е. их аналитическая структура не может быть использована для организации поиска оптимального решения (HP',HW',φ').
4. Критерии оценки качества решения f_i являются противоречивыми. Улучшение одного критерия, как правило, приводит к обязательному ухудшению другого.
5. Для любого из динамических критериев пространство решений имеет большое число локальных минимумов и максимумов, точки разрыва и изолированные точки, не один, а некоторое множество глобальных оптимумов.

В данной работе исследуется возможность и эффективность применения генетических алгоритмов для решения конкретного варианта постановки задачи синтеза архитектур: синтез однородной полносвязной архитектуры оптимальной по числу процессоров и синтез для нее прикладной программы, выполняющейся за заданный директивный срок.

1. Формальная постановка задачи синтеза архитектур ВС оптимальных по числу процессоров. Задача синтеза однородной полносвязной архитектуры оптимальной по числу процессоров и параллельной прикладной программы, выполняющейся за заданный директивный срок на данной архитектуре может быть сформулирована следующим образом:

Для заданных:

- модели поведения прикладной программы H,
- директивного срока ее выполнения (T^{dir}), *требуется определить:*

- M — число процессоров в ВС,
- $HP = \{SP_i\}_{i=(1...M)}$ — привязку процессов, составляющих программу H, к процессорам ВС (SP_i - упорядоченный список рабочих интервалов процессов, назначенных на один процессор),

при этом должны выполняться условия:

1. Время выполнения программы не должно превышать директивный срок $T^w \le T^{dir}$,
2. Число процессоров M в ВС должно быть минимально необходимым для выполнения условия 1.

Модель поведения прикладной программы H задается наиболее сложной историей выполнения программы. H представляет собой ациклический ориентированный размеченный граф: $H = (P, \prec)$. Вершинам $P = \{p_i\}_{i\in(1...N)}$ соответствуют рабочие интервалы процессов [1], дугам $\prec = \{\prec_{ik} = (p_i, p_k)\}_{(i,k)\in(1...N)}$ - связи, определяющие взаимодействия между рабочими интервалами из множества P. Под рабочим интервалом понимаем интервал времени, в течении которого процесс не обращается к другим процессам системы. Все рабочие интервалы одного процесса должны быть назначены на один и тот же процессор. Чередование рабочих интервалов процессов, назначенных на один и тот же процессор, допустимо, если не нарушается частичный порядок, заданный $\prec$. Отношение $\prec_{ik}$ представляется следующим образом: если $p_i \prec_{ik} p_k$, то рабочий интервал p_i необходимо выполнить перед началом рабочего интервала p_k . Каждая вершина имеет свой уникальный номер и метки: принадлежности рабочего интервала к процессу и вычислительной сложности рабочего интервала. Вычислительную сложность рабочего интервала будем задавать временем его выполнения на процессоре. Затраты на взаимодействие будем включать во время выполнения.

Архитектура HW представляет собой полносвязный граф, вершинами которого являются процессоры, дуги – связи между ними. Время выполнения T^W программы HP на архитектуре HW определяется интерпретацией графовых структур HP над структурами HW.

2. Алгоритм синтеза архитектуры ВС. Алгоритм, используемый в синтезе архитектуры ВС РВ, базируется на алгоритме предложенном Холландом и называемом Simple Genetic Algorithm (SGA) [3]. Данный алгоритм можно схематично описать следующим образом:

```
Алгоритм синтеза ( )
[
  задание начальной популяции;
  вычисление критерия выживаемости
  пока не достигнут критерий останова;
  [
    селекция;
    скрещивание;
    мутация;
    вычисление критерия выживаемости;
  ]
]
```

Кодирование. Схематично выбранный способ кодирования может быть представлен следующим образом:

$$<\text{битовая строка}> \equiv (\bigcup_{i=1}^{K} <\text{поле процесса}>_i),$$

<поле процесса> ≡(<номер процессора> $\bigcup$ *<приоритеты интервалов>),*

$$<\text{приоритеты интервалов}> \equiv (\bigcup_{j=1}^{I_i} <\text{приоритет интервала}>_j),$$

где $\bigcup$ - операция конкатенации (склейки) битовых строк, K - число процессов в программе, I_i - число рабочих интервалов в i-ом процессе , $N = \sum_{i=1}^{K} I_i$ - число вершин в графе H.

Поле “номер процессора” однозначно определяет распределение процессов по процессорам. Поле “приоритеты интервалов” используется для определения строгого порядка реализации рабочих интервалов на процессорах [2,4]. Одновременно с определением порядка осуществляется оценка времени выполнения прикладной программы. Выбранный способ кодирования позволяет исключить появление недопустимых конфигураций в результате выполнения операций скрещивания и мутации [2,4].

Критерии останова и выжываемости. Целевая функция позволяет оценивать решение комплексно по набору критериев. В качестве таких критериев в рассматриваемой задаче используются: число процессоров в ВС и время выполнения прикладной программы (время выполнения прикладной программы не должно превышать директивный срок). Пусть k_t —оценка решения по времени выполнения программы, $k_м$ —оценка решения по числу процессоров в ВС. Введем целевую функцию в следующем виде:

$$F(k_t, k_M) = C1\, k_t + C2\, k_M, \quad C1 + C2 = 1$$

где $C1$ и $C2$ - коэффициенты.

Аргументы k_t и $k_м$ принадлежат интервалу (0, 1] и вычисляются следующим образом:

$$k_t = \begin{cases} T^{dir} \big/ T^{w}, & \text{при } T^{dir} \le T^{w} \\ 1, & \text{при } T^{dir} > T^{w} \end{cases},$$

$$k_M = 1 - \frac{M}{K},$$

где K - максимально возможное число процессоров (равно числу процессов в программе), M - число процессоров, использованных в решении.

Для рассматриваемой постановки задачи всегда будет выполняться неравенство:

$$M \ge \left\lceil \sum_{i=1}^{K} \sum_{j=1}^{I_i} t_i \Big/ T^{dir} \right\rceil, \qquad (1)$$

где $\rceil \cdot \lceil$ - округление до ближайшего большего целого и I_i - число рабочих интервалов в i-ом процессе. Если алгоритм находит значение M, для которого выполняется строгое равенство, то оно будет гарантированно оптимальным. Следует отметить, что варианта решения задачи, для которого выполнятся строгое равенство в (2) может вообще не существовать.

Критерий останова зададим следующим образом. Если алгоритм при соблюдении ограничений на директивные сроки находит число процессоров, которое удовлетворяет строгому равенству в (1), то происходит его останов. В противном случае, алгоритм останавливается, если за заданное число итераций, максимальное значение целевой функции улучшено не было.

Селекция. В данном алгоритме используются комбинация схемы пропорциональной селекции и схемы рулетки [3,5]. Для вычисления целого числа потомков используется схема пропорциональной селекции, а для распределения остатка – схема рулетки.

Скрещивание и мутация. В данном алгоритме используется операция скрещивания SGA: одноточечное скрещивание со случайным выбором битовых строк для скрещивания и точки скрещивания.

Для выполнения операции мутации был предложен и реализован следующий алгоритм [4]: вместо того, чтобы вычислять мутацию для каждого бита, вычислялось расстояние от текущего бита. Бит, отстоящий на вычисленное расстояние, обязательно инвертировался.

3. Исследование алгоритма. Исследование алгоритма проводилось на графах поведения прикладных программ H(PR) представляющих собой [2,4]: набор процессов с произвольными информационными связями; набор независимых процессов; набор цепочек процессов; входящее дерево; выходящее дерево; граф адаптивных методов цифровой обработки сигналов [6]. Число вершин в графах изменялось от 15 до 600. Число рабочих интервалов у процессов изменялось от 1 до 8. Дисперсия времен выполнения рабочих интервалов изменялась в пределах: 0÷50%; 50÷100%; 100÷200%. Число итераций алгоритма без улучшения коэффициента выживаемости: 1000÷1500. В качестве начальной популяции во всех тестах задавались два варианта распределения процессов по процессорам:

- все процессы выполняются на одном процессоре;
- каждый процесс выполняется на своем процессоре.

Число битовых строк в начальной популяции с этими вариантами распределения составляло по 50%.

Испытания разработанного алгоритма синтеза структур ВС показали, что его параметры могут быть настроены независимо от типа структуры связей и числа вершин в графе поведения программы, а также дисперсии времен выполнения рабочих интервалов. Оптимальные значения параметров алгоритма находятся в интервалах:

- порог вероятности мутации 0.0075÷0.0085;
- порог вероятности скрещивания 0.3÷0.5;
- весовые коэффициенты (C_1) 0.3÷0.5;5
- размер популяции 25.

Запуск алгоритма с другими значениями параметров значительно ухудшает качество получаемых результатов. Подходящее решение находилось за 50÷1500 итераций алгоритма. Зависимость числа итераций от качества получаемых решений и параметров алгоритма не прослеживается. Например, для входящего дерева с числом вершин 25 хорошие решения получены за 122, 238, 282, 300,362, 407, 481, 788, 826, 1009, 1444, 1481 итераций при различных комбинациях параметров алгоритма в пределах указанных выше интервалов (размер популяции-25).

Наибольшее влияние на сложность получения и качество решения оказывает порог вероятности мутации. Для большинства тестов основное число хороших решений алгоритм получает при пороге вероятности мутации равной 0.008. Очень

сильное влияние на качество получаемых решений оказывает размер популяции. Он имеет ярко выраженный оптимум равный 25.

Заключение. Рассмотренный в данной работе конкретный вариант задачи синтеза архитектур обладает всеми отмеченными во введении особенностями. При увеличении числа оптимизируемых параметров, ограничений, критериев оценки качества решений и уровня детализации моделей, задача синтеза архитектур не приобретает никаких новых особенностей, которые могли бы вывести ее из данного класса задач оптимизации. Это позволяет надеяться, что разработанный генетический алгоритм может быть успешно применен для решения задачи синтеза архитектур и в других ее постановках при добавлении в битовую строку полей, определяющих соответствующие оптимизируемые параметры.

ЛИТЕРАТУРА

1. *Смелянский Р.Л.* Модель функционирования распределённых вычислительных систем// Вестник МГУ, сер.15, Вычислительная математика и кибернетика. - 1990. - Вып.3.
2. *Костенко В.А.* Возможности генетических алгоритмов для решения задач синтеза архитектур и планирования параллельных вычислений // Труды Третьей Международной научной конференции “Дискретные модели в теории управляющих систем” (Красновидово’98). - М.: Диалог МГУ, 1998. - С.53-58.
3. *Holland J.N.* Adaptation in Natural and Artificial Systems. Ann Arbor. -Michigan: Univ. Michigan Press, 1975.
4. *Костенко В.А., Смелянский Р.Л., Трекин А.Г.* Тезисы докладов Всероссийской научной конференции «Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов» (сентябрь 1998 г. г. Новороссийск) М.: Издво МГУ, 1998. С. 5-41.
5. Zbignev Michalewiz. Genetic Algorithms + Data Structures = Evolution Programs. Third revised and extended edition — Springer, 1999.
6. *Костенко В.А.* Крупноблочный параллелизм в задачах обработки сигналов //Программирование. - 1997. - Вып.2.

УДК 658.512.2.0011.5:338.242

Н.А. Семёнов, А.В. Грецкий

О ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ПРИ (РЕ)ИНЖИНИРИНГЕ БИЗНЕС-ПРОЦЕССОВ

Центральным понятием инжиниринга является понятие бизнес-процесса (БП), определяемого как "структурированный, измеряемый набор действий, созданный, чтобы произвести определённый выход для конкретного клиента или рынка" [1]. Становление понятия "бизнес-процесс" отвечает переходу от ориентации на традиционные иерархические структуры при организации производства к горизонтальным (сетевым) связям. На современном этапе принципиально важным становится интеграция в бизнес-процесс, с одной стороны, широкого множества клиентов с *учитываемыми* индивидуальными потребностями, а, с другой стороны, широкого множества поставщиков и сторонних подрядчиков (т.н. аутсорсинг).

Структура современного бизнес-процесса становится всё более разнородной: её составляют клиенты, поставщики, сторонние подрядчики (т.н. внешние субъекты БП) и производственные элементы - рабочие, служащие, оборудование, инфор-