GARNIK MANUKYAN
Ural Federal University named after the first President of Russia BN Yeltsin
Graduate student of the chair of Modeling of Controlled Systems of the Graduate School of Economics and Management
GOHARIK PETROSYAN
International Scientific-Educational Center of National Academy of NAS RA, Lecturer
Candidate of Mathematics, Associate Professor

ABOUT SOME MATHEMATICAL MODELS OF DELIVERY MANAGEMENT

The paper is devoted to the construction and validation of the adaptive mathematical model of delivery management. A mathematical optimization method development of transport routing was realized, taking into account various restrictions and dynamic changes in the transport situation. The construction of routing models has been substantiated and implemented in the form of optimization objective.

ГАРНИК МАНУКЯН
Российская Федерация, Уральский федеральный университет имени первого Президента России Б. Н. Ельцина, аспирант кафедры моделирования управляемых систем Высшей школы экономики и менеджмента

НЕКОТОРЫЕ КОНСТРУКТИВНЫЕ КЛАССИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА

Статья посвящена исследованию современных методов для решения задач маршрутизации транспорта. Приведен краткий обзор основных конструктивных классических алгоритмов, подробно описаны данные алгоритмы и их расширения. Рассмотрена оптимизация отдельного маршрута, эффективность которой подтверждена результатами вычислительного эксперимента, формализация задачи и разработка конкретных алгоритмов, которые в дальнейшем могут быть применены на практике. Также в статье приводится обзор некоторых численных методов решения транспортных задач.

Ключевые слова*: задача маршрутизации транспорта, перенос вершин, обмен рёбер между маршрутами, сбережение, время исполнения.*

Проблема оптимизации поставок продукции всегда привлекала повышенное внимание экономистов, специалистов по логистике и математиков. Среди экономистов это обусловлено важностью транспортировки товаров и связанных с ней издержек (Бауэрсокс, Клосс, 2008, с. 47). Для математиков этот интерес вызван ее значительной сложностью. В литературе рассматриваемая задача известна как задача маршрутизации транспорта (Vehicle Routing Problem); ее не следует путать с транспортной задачей (иногда называемой задачей Монжа-Канторовича), которой посвящено много книг и работ, в том числе известная монография (Гольштейн, Юдин, 1969).

Задача маршрутизации транспорта (ЗМТ) является обобщением известной задачи коммивояжера и, следовательно, принадлежит к классу NP-полных

задач. Это означает, что для нее не найден алгоритм решения за полиномиальное время и не доказано, что такого алгоритма не существует.

1. Классические алгоритмы и их расширения

К настоящему моменту известно достаточно много алгоритмов для решения ЗМТ. Большей частью это эвристические методы, используемые при наличии матрицы расстояний или информации о расположении вершин на плоскости. Известные подходы обычно ориентируются на общую формулировку ЗМТ, в которой предполагается симметричная или несимметричная матрица расстояний, не заданное жёстко количество транспортных средств, и отслеживается только ограничение по их грузоподъёмности или максимальной длине маршрута. В описаниях известных алгоритмов, приводимых ниже, указываются дополнительные уточнения постановки задачи в тех местах, где это важно.

1.1 Алгоритм Кларка-Райта

Алгоритм Кларка-Райта (Clarke and Wright) [1] — это один из самых известных алгоритмов для решения ЗМТ. Его идея основана на процессе слияния мелких маршрутов в более крупные, проводимого до тех пор, пока есть возможность уменьшить суммарную стоимость объезда. Особую роль в этом алгоритме играет понятие "сбережения" (saving) — это снижение общей стоимости решения, получаемое при объединении двух маршрутов. Рассмотрим ситуацию, когда маршрут $(0, \dots, i, 0)$ и маршрут $(0, j, \dots, 0)$ могут быть совмещены в единую последовательность $(0, \dots, i, j, \dots, 0)$. Сбережением является изменение расстояния, равное $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, если оно больше нуля, где c_{ij} — расстояние между соответствующими вершинами. Алгоритм применяется в случаях, когда количество экипажей не определено заранее и его можно вычислять в ходе работы. Его можно использовать как для симметричных задач, так и для несимметричных, но в [2] показано, что качество работы для симметричных случаев заметно ухудшается. Известны два варианта реализации: параллельный и последовательный. В обоих случаях для них требуется предварительное выполнение подготовительного этапа. Он заключается в:

1. Вычислении сбережений $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, *для* $i, j = 1, \dots, n$ *и* $i \neq j$.
2. Создании n маршрутов транспортных средств $(0, i, 0)$ для $i = 1, \dots, n$.
3. Сортировке сбережений в порядке убывания.

Рассмотрим сначала параллельный вариант алгоритма:

1. Просматриваем построенный список сбережений с его начала и выполняем следующие действия:

a) для текущего элемента списка s_{ij} определяем, существуют ли два маршрута, один из которых содержит дугу или ребро $(0, j)$, второй — дугу или ребро $(i, 0)$, и которые могут быть соединены в один общий маршрут;

b) если такие маршруты найдены, то выполняем их объединение, удаляя $(0, j)$ и $(i, 0)$ а затем добавляя (i, j).

Последовательный вариант можно представить следующим образом:

1. Для всех маршрутов $(0, i, \dots, j, 0)$ выполним следующие действия:

a) проведём поиск элемента в списке сбережений s_{ki} или s_{jl}, который может быть использован для объединения текущего маршрута с некоторым, содержащим дугу (ребро) $(k, 0)$ или $(0, l)$;

b) если элемент был найден, то выполним объединение и применим процедуру ещё раз к новому маршруту;

c) если сбережение найти не удалось, то перейдём к следующему маршруту и продолжим работу;

d) завершим работу, когда объединения более невозможны.

Вычислительные результаты сравнения показывают, что параллельный вариант алгоритма даёт результаты лучше, чем последовательный.

1.2 Расширения алгоритма Кларка-Райта

Один из недостатков алгоритма Кларка-Райта заключается в том, что эффективность его работы падает по мере приближения к концу вычислений, в то время как в начале работы решения получаются относительно удачные. В [3] и в [4] предлагалось обобщение понятия сбережения с целью устранения этого недостатка. Сбережение представляется в виде $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$, где λ — параметр для учёта формы маршрута, который позволяет сделать акцент на расстояния между вершинами для соединения. В [5] показано, что $\lambda = 0{,}4$ улучшает решения, как с точки зрения суммарного расстояния, так и с точки зрения конечного количества маршрутов.

Алгоритм Кларка-Райта может оказаться неэффективным с точки зрения времени исполнения, т. к. во всех вариантах все выигрыши должны быть вычислены, сохранены и отсортированы. Различные авторы предпринимали попытки предложить расширения, уменьшающие время исполнения и использование памяти, необходимые для работы этого алгоритма. Большинство из них выполнено в 70-е годы, а также в начале 80-х годов, когда многие исследователи ещё работали с вычислительной техникой, значительно менее производительной, чем в наши дни.

При реализации стратегии, основанной на понятии сбережения, необходимо уделять внимание двум важным аспектам: методу нахождения элемента наибольшего сбережения в списке и требованиям к хранению данных в памяти. Здесь поиск наибольшего сбережения — это самый трудоёмкий процесс. Для решения этой задачи применяются три подхода:

1. Использование полного алгоритма сортировки. Например, сортировки Хоара.
2. Использование ограниченной сортировки при помощи построения кучи [5]. Здесь кучей называется специальное двоичное дерево, в котором сбережения расположены так, что родительский узел всегда имеет большее значение, чем дочерние. Когда происходит слияние маршрутов, то куча позволяет быстро и эффективно исключать соответствующие узлы.
3. Использование специального итеративного алгоритма для вычисления максимального сбережения [6]. В работе было показано, что $s_{ij} > \overline{s}$ всегда,

когда $c_{0i} > 0.5\bar{s}$ и $c_{0j} > 0.5\bar{s}$, на основе того, что все расстояния положительные и выполняется правило треугольника в правой части определения сбережения, где $\bar{s}$ — текущее максимальное значение сбережения. Приведённое необходимое условие затем используется для эффективного нахождения максимального сбережения.

В [7] и в [8] описана ещё одна интересная модификация алгоритма, основанного на понятии сбережения. На каждом шаге сбережение s_{pq}, полученное при слиянии маршрутов p и q, вычисляется как $s_{pq} = t(S_p) + t(S_q) - t(S_p \cup S_q)$, где S_k— множество вершин маршрута k, а $t(S_k)$— длина оптимального решения ЗК, полученного на множестве S_k. Значение $t(S_k)$ может определяться разными способами. Один из них предложен на основе использования приближённой оценки длины точного решения.

Алгоритм Кларка-Райта утратил своё значение как самостоятельного подхода, т. к. предложено много более эффективных методов, но часто используется как способ нахождения начального решения для последующего улучшения.

1.3 Последовательный алгоритм вставки Моля-Джеймсона

Алгоритм Моля-Джеймсона (Mole and Jameson) может применяться для задач с неопределённым заранее количеством транспортных средств. Он использует при расширении маршрута два параметра — λ и μ.

$$\alpha(i,k,j) = c_{ik} + c_{kj} - \lambda c_{ij},$$
$$\beta(i,k,j) = \mu c_{0k} - \alpha(i,k,j),$$

Работа алгоритма может быть представлена следующим образом:

1. Если нет неиспользованных вершин, завершаем работу алгоритма. В противном случае создаём новый маршрут $(0,k,0)$, где k — любая не использованная вершина.

2. Вычисляем для любой неиспользованной вершины k стоимость возможной вставки в новый маршрут $\alpha^*(i_k,k,j_k) = \min\{\alpha(r,k,s)\}$, где r и s — любые две соседние вершины, принадлежащие последнему созданному маршруту. Вершины i_k и j_k— две вершины, соответствующие α^*. Если возможных вариантов вставки не существует, возвращаемся на шаг 1. В противном случае вершину для вставки k^* выберем такую, чтобы она давала $\beta^*(i_{k^*},k^*,j_{k^*}) = \max\{\beta(i_k,k,j_k)\}$ среди всех возможных k. Добавим k^* между вершинами i_{k^*} и j_{k^*}.

3. После добавления вершины выполним оптимизацию при помощи процедуры 3-опт [9] и вернёмся на шаг 2.

Правила вставки контролируются параметрами λ и μ. Например, если $\lambda = 1$ и $\mu = 0$, алгоритм вставит вершину, позволяющую получить минимальное расстояние. Если $\lambda = \mu = 0$, то вставляемая вершина соответствует минимальной сумме расстояний между двумя соседними вершинами. Если $\mu = \infty$ и $\lambda > 0$, то вставляется вершина, максимально удалённая от депо.

1.4 Последовательный алгоритм вставки Кристофидеса-Мингоззи-Тосса

Кристофидес, Мингоззи и Тосс (Christofides, Mingozzi and Toth) разработали более глубокий последовательный алгоритм вставки [10]. Он также применяется для случаев с неопределённым заранее числом транспортных

средств. Это двухфазовый эвристический метод, которым также можно управлять при помощи двух параметров λ и μ.

Первая фаза работы:

1. Зададим номер первого маршрута $k = 1$.

2. Выберем любую неиспользованную вершину i_k для начальной инициализации маршрута k. Для каждой неиспользованной вершины i вычислим $\delta_i = c_{0i} + \lambda_{ii_k}$.

3. Пусть $\delta_{i^*} = \min_{i \in Sk}\{\delta_i\}$, где S_k — множество всех неиспользованных вершин, которые могли бы быть добавлены к маршруту k. Добавим вершину i^* к маршруту k и выполним оптимизацию при помощи процедуры 3-опт [9]. Будем повторять этот шаг до тех пор, пока существуют вершины, которые могли быть добавлены в маршрут k.

4. Если все вершины были использованы в маршрутах, то завершим работу алгоритма. В противном случае установим $k = k + 1$ и вернёмся на шаг 2.

Вторая фаза работы:

1. Создадим k маршрутов $R_t = (0, i_t, 0)$ $(t = 1, \ldots, k,$ где k — количество маршрутов, полученное в конце первой фазы работы. Пусть $J = \{R_1, \ldots, R_k\}$.

2. Для каждого маршрута $R_t \in J$ и для каждой вершины i, ещё не ассоциированной с маршрутом, вычислим $\varepsilon_{ti} = c_{0i} + \mu c_{ii_t}$ и $\varepsilon_{t'i} = \min_{t^*}\{\varepsilon_{ti}\}$. Припишем вершину i маршруту R_{t^*} и будем повторять этот шаг до тех пор, пока все вершины не будут принадлежать маршруту.

Возьмём любой маршрут $R_t \in J$ и установим $J \coloneqq J\{R_t\}$. Для каждой вершины i, ассоциированной с маршрутом R_t, вычислим $\varepsilon_{t'i} = \min_{R_t \in J}\{\varepsilon_{t_i}\}$ и $\tau_i = \varepsilon_{t'i} - \varepsilon_{ti}$.

3. Добавим в маршрут R_t вершину i^*, соблюдая $\tau_{i*} = \max_{i \in S_t}\{\tau_i\}$, где S_t — множество неиспользованных вершин, ассоциированных с маршрутом R_t, которые могли бы быть добавлены в маршрут R_t. Выполним оптимизацию маршрута R_t при помощи процедуры 3-опт. Будем повторять этот шаг до тех пор, пока есть вершины, которые можно добавить в маршрут R_t.

4. Если $|J| \neq \emptyset$, то возвращаемся к шагу 2 второго этапа. В противном случае, если неиспользованных вершин больше нет, то завершаем работу. Если есть неиспользованные вершины, то создаём новые маршруты, начиная с шага 1 первого этапа.

2. Классические улучшающие алгоритмы

Улучшающие алгоритмы для ЗМТ обрабатывают либо отдельный маршрут за раз, либо несколько маршрутов. В случае одного маршрута можно применять любые алгоритмы оптимизации для ЗК. Для второго варианта могут быть разработаны алгоритмы, которые анализируют структуру, представленную несколькими маршрутами.

2.1 Оптимизация отдельного маршрута

Большинство процедур оптимизации для ЗК могут быть описаны в терминах λ −опт операций, которые были предложены Лином [9]. В них λ рёбер

удаляются из маршрута, и λ оставшихся сегментов пересоединяются во всех возможных комбинациях. Если улучшающее соединение найдено (первое достигнутое или наиболее удачное), то изменения вносятся в маршрут. Работа останавливается на локальном минимуме, если более невозможно найти подходящие варианты замен. Проверка λ −оптимальности решения требует времени $O(n^{\lambda})$.

Предложены несколько вариантов этого подхода. Лин и Керниган [12] представили алгоритм, в котором λ изменяется динамически в процессе поиска. В [13] описан метод *Or*-опт, заключающийся в подмене последовательностей из 3, 2 или 1 соседних вершин на последовательность из другого фрагмента этого же маршрута. Фактически он представляет собой ограниченную форму 3-опт оптимизации. Проверка *Or*-оптимальности требует времени $O(n^2)$. На той же основе в [11] предлагается ограниченная форма 4-опт алгоритма, называемая 4-opt* и требующая для проверки оптимальности времени $O(\omega n^2)$.

В [14] проведён анализ упомянутых методов, где авторы пришли к заключению, что оптимизация Лина-Кернигана даёт в среднем самые хорошие результаты.

2.2 Алгоритмы для улучшения нескольких маршрутов

В [15, 16, 17] приводятся алгоритмы, основанные на обмене рёбер между маршрутами. Они вобрали в себя различные схемы, предложенные ранее несколькими авторами [18, 19, 20].

Особое внимание уделяется общей схеме *b*-циклического *k*-переноса, в которой рассматривается циклическая перестановка b маршрутов, и выполняется перенос k вершин из каждого маршрута в следующий маршрут. Авторы показывают, что применение некоторых обменов *b*-циклического *k*-переноса (с $b = 2$ или с переменным значением b и $k = 1$ или $k = 2$) даёт интересные результаты.

В [15] перечислены основные варианты модификации маршрутов, они являются частными случаями 2-циклических переносов.

1. Перекрещивание двух рёбер из двух маршрутов (рис. 1).
2. Обмен вершинами между двумя маршрутами (рис. 2).
3. Перенос вершин из маршрута в маршрут (рис. 3).
4. Комбинации предыдущих вариантов.

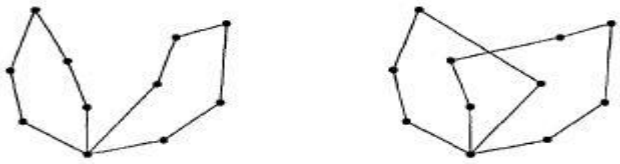

Рис. 1. *Пересечение двух рёбер двух маршрутов*

Рис. 2. *Обмен вершинами двух маршрутов*

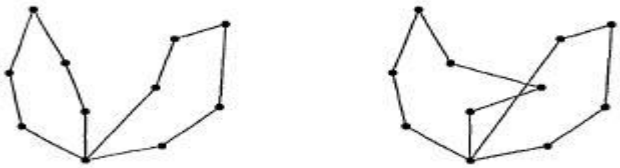

Рис. 3. *Перенос вершин из маршрута в маршрут*

3. Заключение

Вышеуказанные алгоритмы решения ЗМТ не требуют большого количества оперативной памяти для вычислений. В задачах, использующих матрицу стоимости переездов, память в основном расходуется на хранение данных о затратах на перемещение транспортных средств. Поскольку алгоритмы на основе матрицы стоимости применяются для обработки задач, содержащих около 1000 вершин, легко подсчитать, что для них потребуется около 8 МБ свободного пространства. Для геометрических задач, которые могут содержать до 1000000 вершин, матрица не требуется, и основной расход приходится на хранение самих вершин, что требует порядка 20 МБ свободного пространства. Приведённые оценки расхода оперативной памяти в десятки раз меньше, чем её объём, доступный в современных вычислительных устройствах, и нет необходимости применять дополнительные оптимизации.

ЛИТЕРАТУРА

1. **Clarke G**. Scheduling of vehicles from a central depot to a number of delivery points / G. Clarke, J.W. Wright // Operations Research. — 1964. — № 12 — P. 568-581.

2. **Vigo D.** A heuristic algorithm for the asymmetric capacitated vehicle routing problem // European Journal of Operational Research. — 1996. — № 89. — P. 108-126.

3. **Gaskell T.J**. Bases for vehicle fleet scheduling // Operational Research Quarterly. - 1967. - № 18. - P. 281-295.

4. **Yellow P**. A computational modification to the savings method of vehicle scheduling // Operational Research Quarterly. — 1970. — № 21. — P. 281-283.

5. **Golden B.L**. Implementing vehicle routing algorithms / B.L. Golden, T.L. Magnanti, H.Q. Nguyen // Networks. — 1977. — № 7. — P. 113-148.

6. **Paessens H**. The savings algorithm for the vehicle routing problem // European Journal of Operational Research. — 1988. — № 34. — P. 336-344.

7. **Desrochers M.** A matching based savings algorithm for the vehicle routing problem / M. Desrochers, T.W. Verhoog // Les Cahiers du GERAD G-89-04, Ecole des Hautes Etudes Commerciales de Montreal, 1989.

8. **Altinkemer K.** Parallel savings based heuristic for the delivery problem / K. Altinkemer, B. Gavish // Operations Research. — 1991. — № 39. — P. 456-469.

9. **Lin S.** Computer solutions of the traveling salesman problem // Bell System Technical Journal. — 1965. - № 44. - P. 2245-2269.

10. Christofides INT The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, C. Sandi, editors/ N. Christofides, A. Mingozzi, P. Toth // Combinatorial Optimization. — Wiley, Chichester, 1979. — P. 315-338.

11. **Renaud J**. A fast composite heuristic for the symmetric traveling salesman problem / J. Renaud, F.F. Boctor, G. Laporte // INFORMS Journal on Computing. - 1996. - № 8. - P. 134-143.

12. **Lin S.** An effective heuristic algorithm for the traveling salesman problem / S. Lin and B. Kernighan // Operations Research. — 1973. — № 21. — P. 498-516.

13. Or. I. Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking. Ph.D. dissertation. — Northwestern University, Evanston, IL, 1976.

14. **Johnson D.S**. The traveling salesman problem: A case study. In E.H.L. Aarts and J.K. Lenstra, editors, / D.S. Johnson, L.A. McGeoch // Local Search in Combinatorial Optimization. — Wiley, Chichester, 1997. — P. 215-310.

15. **Thompson P.M**. Cyclic transfer algorithms for the multivehicle routing and scheduling problems / P.M. Thompson, H.N. Psaraftis // Operations Research. — 1993. — № 41. - P. 935-946.

16. **Van Breedam A.** An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints. Ph.D. dissertation. — University of Antwerp, 1994.

17. **Kinderwater G.A.P.** Vehicle routing: Handling edge exchanges. In E.H.L. Aarts, J.K. Lenstra, editors / G.A.P. Kinderwater and M.W.P. Savelsbergh // Local Search in Combinatorial Optimization. — Wiley, Chichester, 1997. - P. 337-360.

18. **Stewart W.R**. A Lagrangean relaxation heuristic for vehicle routing / W.R. Stewart Jr and B.L. Golden // European Journal of Operational Research. - 1984. - № 15. - P. 84-88.

19. **Dror M.** A vehicle routing improvement algorithm. Comparison of a 'Greedy' and a 'Matching' implementation for inventory routing / M. Dror, L. Levy // Computers *к* Operations Research. — 1986. — № 13. — P. 33-45.

20. **Salhi S.** Improvements to vehicle routing heuristics / S. Salhi, G.K. Rand //Journal of the Operational Research Society. — 1987. — № 38. — P. 293-295.

ԳԱՌՆԻԿ ՄԱՆՈՒԿՅԱՆ

Ռուսաստանի առաջին նախագահ Բ. Ն. Ելցինի անվան Ուրալի դաշնային համալսարան, Տնտեսագիտության և կառավարման բարձրագույն դպրոցի Կառավարվող համակարգերի մոդելավորման ամբիոնի ասպիրանտ

ՏՐԱՆՍՊՈՐՏԱՅԻՆ ԽՆԴՐԻ ԼՈՒԾՄԱՆ ՈՐՈՇ ԿԱՌՈՒՑՈՂԱԿԱՆ ԴԱՍԱԿԱՆ ԱԼԳՈՐԻԹՄՆԵՐ

Հոդվածը նվիրված է տրանսպորտի երթուղավորման խնդրի ժամանակակից մեթոդների հետազոտմանը: Բերված են հիմնական դասական կառուցողական ալգորիթմների կարճ ակնարկը, մանրամասնորեն նկարագրված են տրված ալգորիթմները և նրանց ընդլայնումները: Դիտարկված է առանձին երթուղու օպտիմալացումը, որի արդյունավետությունը հաստատված է հաշվողական փորձի արդյունքներով, խնդրի ձևակերպումը և կոնկրետ ալգորիթմների մշակումը, որոնք հետագայում կարող են կիրառվել

պրակտիկայում: Ինչպես նաև հոդվածում բերված է տրանսպորտային խնդիրների լուծման որոշ թվային մեթոդների ակնարկը:

GARNIK MANUKYAN
Ural Federal University named after the first President of Russia BN Yeltsin,
Graduate student of the chair of Modeling of Controlled Systems of the Graduate School of Economics and Management

SOME CONSTRUCTIVE CLASSICAL ALGORITHMS FOR SOLVING THE PROBLEM OF VEHICLE ROUTING

The article is dedicated to the study of modern methods for solving the problems of vehicle routing. A brief review of the main constructive classical algorithms is provided and these algorithms and their extensions are described in detail. The optimization of a single route is considered, the effectiveness of which is confirmed by the results of a computational experiment, the formalization of the problem and the development of specific algorithms that can later be applied in practice. Also, the article provides an overview of some numerical methods for solving transport problems.

ՄԵՐՈՒԺԱՆ ՄԱՐՏԻՐՈՍՅԱՆ
Սինոփսիս Արմենիա,
Համալսարանական ծրագրերի մասնագետ

ՇԵՂՈՒՄՆԵՐԻ ՉԵՎԱՉԱՓԻ ՍՏԱՑՈՒՄԸ ՕԳՏԱԳՈՐԾԵԼՈՎ 14 ՆԱՆՈՄԵՏՐԱՆՈՑ ՏՐԱՆԶԻՍՏՈՐՆԵՐԻ ՄՈՆՏԵ ԿԱՐԼՈ ՄՈԴԵԼՆԵՐԸ

Գործընթացի ու միջավայրի պարամետրերը չեն կարող նույնը լինել ինտեգրալ սխեմայի(ԻՍ) տարբեր հատվածներում: Գործընթացի շեղումների պատճառով առանձին տրանզիստորները ԻՍ-ի տարբեր հատվածներում ունեն տարբեր բնութագծեր: Այս տարբերությունը առաջանում է ԻՍ-ում գործընթացի շեղումների տարբերություններից: Վերջինս իր մեջ ներառում է երկու տեսակի շեղումներ՝ տեղային և ընդհանուր: Տեղային գործընթացի շեղումը տեղի է ունենում մեկ ԻՍ-ում, իսկ ընդհանուրը՝ տարբեր ԻՍ-երում, որի գործընթացի շեղումը շատ անգամ ավելի մեծ է քան տեղայինինը: Այս ուսումնասիրության մեջ օգտագործվում է տեղային գործընթացի շեղումը:

Բանալի բառեր՝ *նոնոմետրանոց տրանզիստորներ, մոնտեկարլո մոդելներ, ինտեգրալ սխեմա, տեխնոլոգիական շեղումներ:*

Ինտեգրալ սխեմա: Բարդ էլեկտրոնային սարքեր (ինչպիսիք են ժամանակակից համակարգիչները, հեռախոսները, հեռուստացույցները և այլն, որոնք պարունակում են մեծաթիվ էլեկտրոնային տարրեր) ստեղծելու համար օգտագործվում է ինտեգրալ սխեման: ԻՍ-ի հիմնական գաղափարը կայանում է նրանում, որ մեծաթիվ (ներկայումս նույնիսկ մի քանի տասնյակ միլիարդը գերազանցող) առանց «պատյանի»էլեկտրոնային բաղադրիչներից և դրանց միջև միջմիացումներից բաղկացած ամբողջական էլեկտրոնային սխեման ներկառուցվում է կիսահաղորդչի (սիլիցիումի կամ գերմանիումի) միկրոսկոպիկ բարակ բյուրեղում, որը տեղակայվում է բոլոր տարրերի համար ընդհա-