УДК 519.6

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МЕТОДА ИЗМЕНЯЮЩИХСЯ ВЕРОЯТНОСТЕЙ

Казаковцев Л.А., Ступина А.А.

ФГОУ ВПО «Сибирский государственный аэрокосмический университет им. ак. М.Ф. Решетнева», Красноярск, Россия (660014, г. Красноярск, просп. им. газеты «Красноярский Рабочий», 31), e-mail: levk@ieee.org

Методы случайного поиска находят применение для решения широкого круга дискретных задач оптимизации большой размерности, когда использование детерминированных методов становится невозможным без упрощения исходной задачи из-за резкого роста требуемых вычислительных мощностей. Даже в случае применения случайного поиска, однако, требуются весьма значительные вычислительные мощности, и точность решения зависит от затраченного времени. Быстрое развитие сравнительно дешевых многопроцессорных систем позволяет значительно сократить время поиска приемлемого решения с коэффициентом ускорения, близким к идеальному. Здесь рассмотрен подход к распараллеливанию алгоритма, реализующего модифицированный вариант метода изменяющихся вероятностей с адаптацией и процедурой возврата для задач оптимизации псевдобулевой функции с ограничениями. Существующий оптимизационный алгоритм адаптирован для систем с общей памятью (использована библиотека GNUOpenMP и соответствующий компилятор). Оценена эффективность параллельного алгоритма.

Ключевые слова: параллельные алгоритмы, комбинаторная оптимизация.

PARALLEL REALIZATION OF THE PROBABILITY CHANGING METHOD

Kazakovtsev L.A. , Stupina A.A.

Siberian State Aerospace University named after M.F. Reshetnev, Krasnoyarsk, Russia (660014, Krasnoyarsk, Krasnoyarski Rabochiav., 31), e-mail: levk@ieee.org

Random search methods are implemented for solving wide variety of large-scale discrete optimization problems when implementing of the determined methods is impossible due to increase of the computational capacity needed. Even the random search methods need sufficient computational facilities and the precision of the solution depends on the dime spent for problem solving. Rapid development of comparatively inexpensive multiprocessor systems allows us to decrease significantly the time spent for obtaining an acceptable solution with almost ideal parallel efficiency coefficient. Here, we consider an approach for parallelizing an algorithm realizing the modified variant of probability changing method with adaptation and rollback procedure for constrained pseudo-Boolean optimization problems. The existing optimization algorithm is adapted for shared memory systems (with use of GNU OpenMp library and corresponding compiler). Parallel algorithm efficiency is estimated.

Key words: parallel algorithms, combinatorial optimization.

Введение

Методы случайного поиска являются эвристическими методами, не гарантируют нахождения точного результата, но являются статистически оптимальными, т.е. процент задач, решенных «почти оптимально», растет с ростом размерности решаемых задач [1].

Задачами большой размерности можно считать задачи с десятками тысяч, иногда миллионами переменных. Например, задача ассортиментного планирования продукции производственного предприятия или оптимизации госзакупок может включать в себя выбор тысяч наименований товаров у сотен поставщиков. Такие задачи, в общем случае, могут быть решены без их упрощения только методами случайного поиска.

Детерминированные методы, базирующиеся на легко распараллеливаемом методе

ветвей и границ, находят свое применение во многих областях [1; 5], но относятся к классу сложности NP-hard и, несмотря на широкий набор способов [9] снижения вычислительной сложности задач путем учета их характерных особенностей, требуют поиска в дереве, число ветвей которого растет экспоненциально с ростом числа переменных, что делает невозможным применение метода при большой размерности задач.

Решаемая задача, примененный метод

В качестве тестового примера рассмотрим следующую задачу.

$$F(X) = (\sum_{i=1}^{N \cdot V} a_i x_i) \sum_{i=1}^{N} (1 - c_i \sum_{j=1}^{V} x_{V(i-1)+j}) \to max; \quad (1)$$

с ограничениями:

$$\begin{cases} \sum_{i=1}^{N \cdot V} b_{i1} x_i \le B_1; \\ \sum_{i=1}^{N \cdot V} b_{i2} x_i \le B_2; \\ \dots \\ \sum_{i=1}^{N \cdot V} b_{iN_{Constr}} x_i \le B_{N_{Constr}}. \end{cases} \quad (2)$$

$$\sum_{j=1}^{V} x_{V(i-1)+j} \le 1 \forall 1 \le i \le N. \quad (3)$$

Здесь $F(X)$ – целевая функция, x_{ij} – булевы переменные, a_{ij}, b_{ijk}, c_i и B_1, B_2, ..., $B_{Nconstr}$ – константы, N, V – натуральные числа. N – число элементов, из которого делается выбор (товаров, узлов, каналов связи и т.д.), V – число возможных вариантов для каждого элемента. Предполагается, что для каждого из N элементов может быть выбрано не более 1 варианта. Переменная $x_{V(i-1)+j}$= 1, если решение включает i-й элемент в его j-м варианте. Целевая функция является мультипликативной сверткой двух линейных критериев. Подобные задачи возникают, например, при ассортиментном планировании, маршрутизации и т.п. [6].

Изначально разработанный для решения задач безусловной оптимизации, метод изменяющихся вероятностей (МИВЕР) – это метод случайного поиска, рассматриваемый иногда как вариант генетического алгоритма, организованный по следующей схеме [1].

1. k=0; устанавливаем начальные значения вероятностей $P_k=\{p_{k1}, p_{k2}, ..., p_{kN}\}$, где $p_{kj}=P\{x_j=1\}$. Установка адекватных начальных значений вектора вероятностей P_k – важная проблема при решении задач данным методом.
2. Согласно вектору вероятностей P_k генерируем набор независимых случайных булевых значений X_{ki}. Значение элемента вектора вероятностей p_{ki} определяет вероятность того, что булева переменная X_{ki} примет значение 1 в генерируемом наборе.
3. Производится расчет значений функции для $F(X_{ki})$ для каждого сгенерированного набора булевых переменных.

4. Выбираются некоторые значения $F(X_{ki})$ и соответствующие им точки X_{ki} (например, выбираются точки, в которых получено минимальное и максимальное значения целевой функции).
5. По результатам пункта 4 модифицируется вектор P_k.
6. k=k+1; если k<R, то переходим к пункту 2. Это условие останова может варьироваться в зависимости от реализации алгоритма.
7. Иначе – останов и результат:

$$F_{min}(X) = \min_{k=1,\dots,R}\{\min_{i=1,\dots,N} F(X_{ki})\}. \quad (4)$$

Модифицированная версия алгоритма [6] позволяет решать задачи с миллионами булевых переменных. При большой размерности даже многократное вычисление значений целевой функции требует значительных ресурсов. В практических задачах (например, в задачах ассортиментного планирования) число ограничений также растет с ростом размерности, растет и число шагов, требуемое для достижения приемлемого результата. Целевая функция иногда задается алгоритмом, который сам представляет собой решение вложенной задачи оптимизации. Требуемые вычислительные ресурсы при этом значительны, и вопрос использования возможностей параллельных систем чрезвычайно актуален.

Последовательная реализация алгоритма

Схема алгоритма для однопроцессорных систем показана на рис. 1, вариант 1.

На шаге инициализации в простейшем случае все компоненты p вектора P устанавливаются равными 0.5, но поскольку алгоритм реализует метод условной оптимизации, начальные значения вектора P важны для попадания генерируемых решений в область допустимых значений. На каждом шаге алгоритма генерируется набор векторов X булевых переменных. Компоненты вектора вероятностей P определяют математическое ожидание генерации значения 1 компонент вектора X. Благодаря ограничениям (3) оптимальные начальные значения компонент вектора P не превышают $1/(V+1)$ [6].

Условием останова является выполнение предельного числа шагов M адаптации вектора P. Может быть также использовано предельное время работы алгоритма или предельное число шагов, не дающих улучшения достигнутого результата.

В алгоритме использована дополнительная штрафная функция [3]:

$$f_{ki} = F_P(X) = C_{PENALTY} \sum_{k=1}^{N_{CONSTR}} F_{Pk}(X); \quad (5)$$

$$F_{Pk}(X) = \begin{cases} 0, & \sum_{i=1}^{N\cdot V} b_{ik}\, x_i \le B_k; \\ \dfrac{\sum_{i=1}^{N\cdot V} b_{ik} x_i - B_k}{B_k}, & \sum_{i=1}^{N\cdot V} b_{ik}\, x_i > B_k. \end{cases} \quad (6)$$

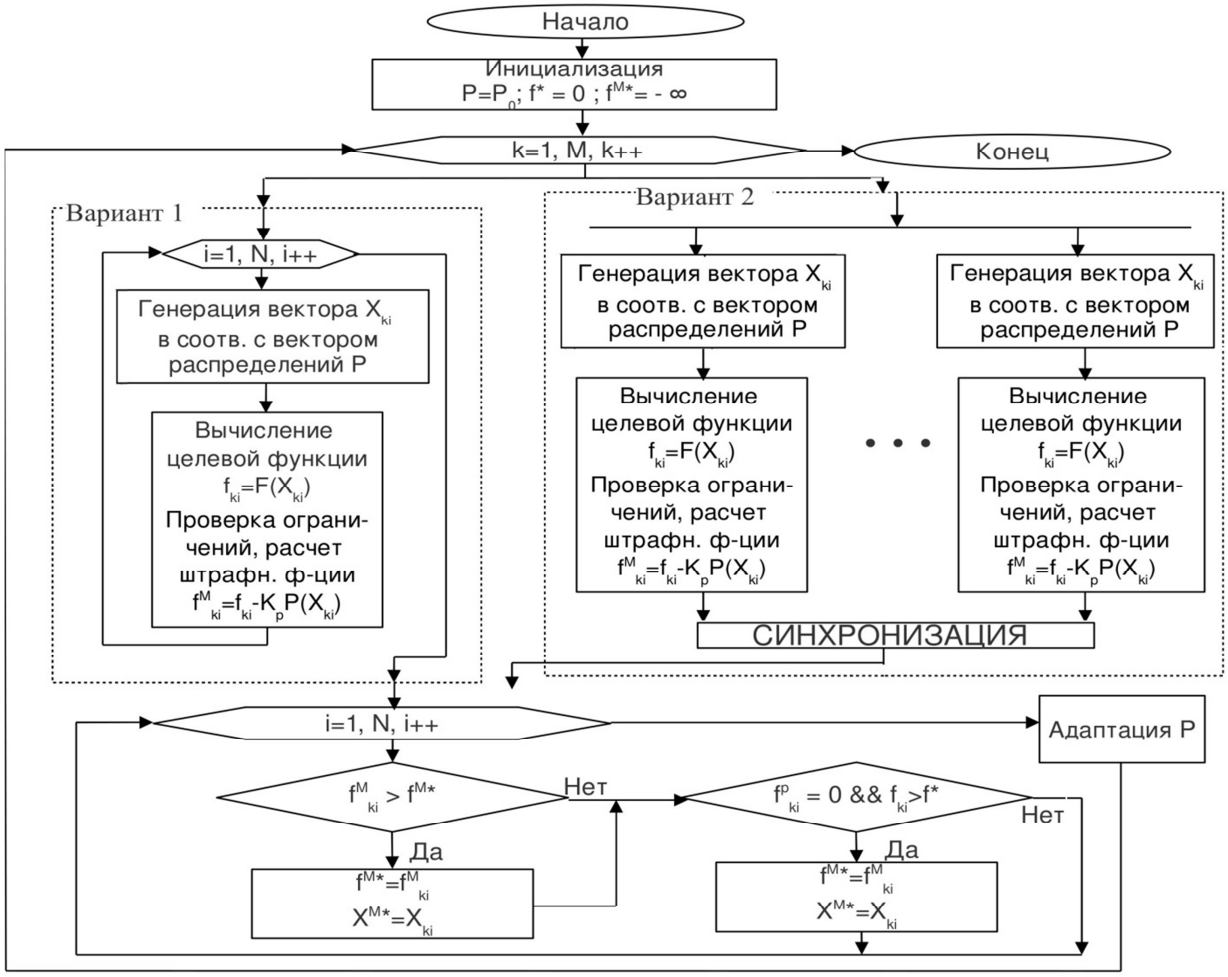

Рис. 1. Блок-схема последовательного (вар. 1) и параллельного (вар. 2) алгоритмов.

Если известна оценка максимально теоретически достижимого значения $f(X)$, мы можем ее использовать в качестве коэффициента $C_{PENALTY}$. В нашем случае[6]:

$$C_{PENALTY} = \sum_{i=1}^{N \cdot V} | a_i |. \quad (7)$$

Модифицированная целевая функция f^M является суммой собственно целевой функции и штрафной функции.

Решение практических задач [6] показывает, что лучшие результаты достигаются при использовании мультипликативной адаптации и процедуры возврата [1].

$$p_{k,j} = \begin{cases} p_{(k-1),j} \cdot d, & x_{kj}^{max} = 1 \wedge x_{kj}^{min} = 0 \wedge p_{(k-1),j} < 0.5, \\ 1 - \frac{(1-p_{(k-1),j})}{d}, & x_{kj}^{max} = 1 \wedge x_{kj}^{min} = 0 \wedge p_{(k-1),j} \geq 0.5, \\ \frac{p_{(k-1),j}}{d}, & x_{kj}^{max} = 0 \wedge x_{kj}^{min} = 1 \wedge p_{(k-1),j} < 0.5, \\ 1 - \left(1 - p_{(k-1),j}\right) \cdot d, & x_{kj}^{max} = 0 \wedge x_{kj}^{min} = 1 \wedge p_{(k-1),j} \geq 0.5, \\ p_{(k-1),j}, & x_{kj}^{max} = x_{kj}^{min}. \end{cases} \quad (8)$$

Здесь d – коэффициент адаптации, который в данном случае от шага k не зависит.

Адаптация вектора P приводит к генерации мало различающихся экземпляров X вокруг локального оптимума. Процедура отката приводит вектор P к начальному (или другому) значению. Условия старта процедуры отката в нашем случае – достижение определенного числа шагов, не улучшающих f^M. Мы выполняем откат по формуле:

$$p_{kj} = (p_{k\text{-}1\,j} + q_k p_0)/(1+q_k), \text{ if } p_{k\text{-}1\,j} < p_0. \quad (9)$$

Здесь p_0 – начальное значение вероятности. Коэффициент q_k может быть постоянным или меняться в зависимости от результатов предыдущих шагов. Например, если s_m – число шагов, сделанных после последнего улучшения найденного оптимума целевой функции,

$$q_k = w / s_m. \quad (10)$$

Для задач оптимизации с тысячами переменных приемлемы значения весового коэффициента w в диапазоне 0,01 ... 0,05.

Процесс минимизации штрафной функции путем адаптации вектора P приближает генерируемые векторы X к области допустимых решений. Процедура отката снова отодвигает нас от этой области. Пусть задача имеет только одно ограничение:

$$b_1 x_1 + b_2 x_2 + \cdots + b_{N \cdot V} x_{N \cdot V} < B. \quad (11)$$

Математическое ожидание левой части выражения (11) равно:

$$M = p_0 b_1 + p_0 b_2 + \ldots + p_0 b_D = (b_1 + b_2 + \ldots + b_D) p_0. \quad (12)$$

Поскольку оптимальные значения целевой функции часто достигаются на границе области допустимых решений ($b_1 x_1 + b_2 x_2 + b_3 x_3 + \ldots + b_D x_D \approx B$), оптимальное начальное значение компонент вектора вероятностей равно:

$$p_0 = B/(b_1 + b_2 + b_3 + \ldots + b_D). \quad (13)$$

В более сложных случаях негативный эффект процедуры отката может быть снижен сохранением среднего значения компонент вектора вероятностей p_0:

$$p_0 = (p_1 + p_2 + \ldots + p_D)/D. \quad (14)$$

Параллельная версия для мультипроцессорных систем

Адаптация алгоритма для систем с общей памятью может быть выполнена путем организации параллельной генерации экземпляров вектора X и их оценки. Схема показана на рис. 1, вариант 2. Каждый из N_P процессоров генерирует по N/Np экземпляров вектора X и вычисляет для них значение $f(X)$, левых частей неравенств-ограничений.

Организация параллельного потока требует значительных ресурсов, эквивалента 1000 операциям деления чисел с плавающей точкой [4], эффективна только при большой размерности.

Для реализации параллельной версии нашего алгоритма требуются лишь

минимальные изменения в исходном коде программы. Если исходный код последовательной версии алгоритма написан на языках C/C++ или Fortran и есть возможность использования библиотеки OpenMP с подходящими компиляторами, требуется лишь вставить дополнительную строку перед началом вложенного цикла. Пример для Фортрана:

```
        do k=1, M
c$omp parallel do
        do i = 1, N
        <Генерация вектора X согласно вектору P>
                <оценка полученных векторов и т.д.>
                enddo
                <Адаптация, возможно — вызов процедуры отката>
        enddo
c$omp end parallel do
```

Также наша программа должна содержать специальные строки в ее начальной части (`$omp` для Фортрана, `#pragmaomp` для C++), которые предписывают компилятору использовать библиотеку OpenMP с нужными опциями.

Работа в системах с распределенной памятью (кластерах)

В [10] предложен подход для генетического алгоритма на системах с распределенной памятью, аналогичный приведенному выше: параллельная генерация векторов X и вычисление целевой функции, что требует интенсивного обмена данными между узлами и частой синхронизации (процедура MPI_BARRIER). Объем информации существенно снижается, если каждый из процессов генерирует множество экземпляров вектора X и отсылает лишь информацию о лучшем и худшем вариантах. Но синхронизация в таких системах требует столько времени и ресурсов, что эффект от параллельного выполнения минимален. На системе из двух компьютеров, соединенных сетью GigabitEthernet (Linux, OpenMPI+ifortran), положительный эффект достигается при размерности задач 1000-3000. При меньшей размерности эффект отрицательный.

Эксперименты на кластере Argo Международного центра теоретической физики Абдус Салам (г. Триест, Италия) с применением библиотеки MPI и компиляторов IntelFortran и GNUFortran также подтверждают низкую эффективность метода. Кластер состоит из узлов на базе процессора IntelXeon 2.6ГГц, соединенных сетью сеть MiryNet. Запуск алгоритма для задачи в 10000 переменных дает незначительный положительный эффект (коэффициент параллельной эффективности 1,23) на 2 узлах, при выполнении на трех и более узлах эффект стабильно отрицательный. Среднее время простоя узлов достигает 43% при 3 узлах.

Результаты для систем с общей памятью

Аналитическая оценка вычислительной сложности в рекуррентных формулах дана в [2], формулы требуют указания точной информации о целевой функции (линейность, унимодальность и т.д.) Сравним затраты времени на выполнение параллельной части алгоритма (рис. 1, вариант 2) и последовательной его части, включающей адаптацию вектора P (линейно зависит от размерности задачи) и возможную процедуру отката (также линейно зависит). При линейной целевой функции (или произведении линейных функций, как в нашем случае) и линейных ограничениях сложность параллельной части также линейно зависит от числа переменных и линейно зависит от числа ограничений, которое часто также линейно зависит от числа переменных (число ограничений (3) напрямую зависит от числа переменных). Сложность параллельной части, таким образом, пропорциональна квадрату числа переменных $O(N^2V^2)$ при линейной сложности последовательной части $O(NV)$, поэтому параллельная эффективность алгоритма растет с ростом размерности.

Экспериментальная оценка эффективности алгоритмов, предложенная в [7], может быть применена и в случае условной оптимизации. Генерируется набор тестовых задач. Коэффициенты целевой в (1-3) устанавливаются генератором случайных чисел [5]. Значения B_i устанавливаются так, чтобы задача имела допустимые решения. Задача решается последовательной версией алгоритма. В качестве условия останова используется максимальное время, отведенное на работу алгоритма. Фиксируется достигнутое значение целевой функции f_G^{M*}. Затем алгоритм запускается K раз для решения той же задачи, но в качестве условия останова принимается достижение того же или лучшего значения f_G^{M*}. Среднее время работы параллельной и последовательной версий сравнивается.

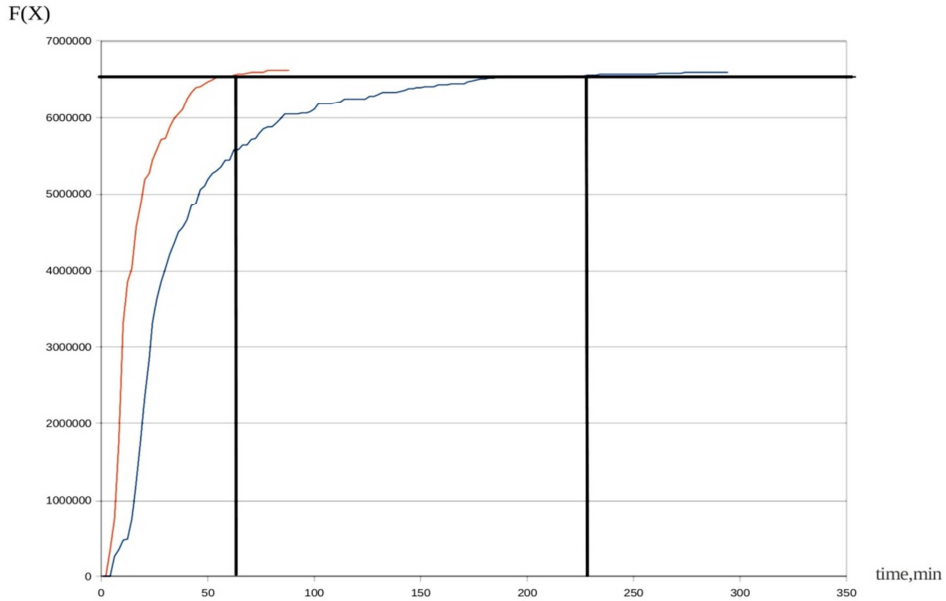

Рис. 2. Результаты последовательного (синий цвет) и параллельного (красный) алгоритмов.

Для оценки параллельной эффективности мы использовали систему с 4 процессорами IntelXeon 2.6 ГГц для решения задач размерностью 1000-50000 переменных. Сравнение среднего значения времени для 5 целевых функций с 10000 переменных, по 10 запусков для каждой, показало параллельную эффективность 0,95-0,97. Для построения графика (рис. 2) в алгоритм включен блок, записывающий результаты и затраченное время через каждые 100 шагов. Для достижения контрольного значения f_G^{M*}=65015.17 последовательной версии понадобилось 13748 сек., параллельной – 3556 сек.

Выводы

Алгоритмы, реализующие метод изменяющихся вероятностей, могут быть легко адаптированы для многопроцессорных систем с высокой параллельной эффективностью. Тот же подход не работает для систем с распределенной памятью, в этом случае требуется более серьезная переработка алгоритма.

Список литературы

1. Антамошкин А.Н. Оптимизация функционалов с булевыми переменными. – Томск : Изд-во Томского ун-та, 1987.

2. Устюжанинов В.Г. О случайном поиске оптимальных систем // Компьютерные системы. Автоматизация разработки в микроэлектронике. Теория, методы, алгоритмы. ИМ СО АН СССР. – 1978. – Вып. 77. – С. 42-51.

3. Baba N. Convergence of Random Search Optimization Methods for Constrained Optimization Methods // Journal of Optimization Theory and Applications. – 1981. – Issue 33. – P. 451-461.

4. Birge J.R., Rosa C.H. Parallel Decomposition of Large-Scale Stochastic Nonlinear Programs, Annals of Operation Research. – 1996. – Issue 64. – P. 39-65.

5. Fishburn P. Utility Theory of Decision Making, J. Wiley. – New York, 1970.

6. Kazakovtsev L. Adaptive Model of Static Routing for Traffic Balance between Multiple Telecommunication Channels to Different ISPs. – URL: http://eprints.ictp.it/415.

7. Ljung L. System Identification: Theory for the User. – Prentice Hall. – NJ, 1987.

8. Pardalos, P. M. Pitsoulis, L. Mavridou, T. Resende, M. G. C. Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP, Lecture Notes in Computer Science. – 1995. – Issue 980. P. 317-331.

9. Ralphs T.K., Ladanyi L., Saltzman M.J. Parallel Branch, Cut and Price for Large-Scale Discrete Optimization in Computational Combinatorial Optimization, D.Naddef and M. Junger, eds. –

Springer. – Berlin, 2001. – P. 223-254.

10. Rinaudoy S., Moschellay F., Anilez M.A. Controlled Random Search Parallel Algorithm for Global Optimization with Distributed Processes on Multivendor CPUs. – URL: http://citeseer.ist.psu.edu/710446.html.

Исследование выполнено при поддержке ФЦП «Научные и научно-педагогические кадры России на 2009-2013 годы».

Рецензенты:

Антамошкин А.Н., д.т.н., проф., зав. кафедрой математического моделирования и информатики ФГОУ ВПО «Красноярский государственный аграрный университет», г. Красноярск.

Петров М.Н., д.т.н., проф., зав. кафедрой электронной техники и телекоммуникаций Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева, г. Красноярск.