5. Консервативность энергетического рынка (smart-системы пока еще не стали необходимостью для российского потребителя);
6. Высокая стоимость оборудования, а также обслуживания и модернизации интеллектуальных систем;
7. Дефицит квалифицированных кадров [2].

ЛИТЕРАТУРА

1. Окороков В.Р., Волкова И.О., Окороков Р.В. Интеллектуальные энергетические системы: технические возможности и эффективность / Академия энергетики. – №2, 2010 .
2. Кобец Борис, Волкова Ирина. SmartGrid / ЭнергоРынок. – №3, 2010.
3. Зибров В.П. Система электроснабжения Псковской области // Отчет учебно-научно-производственного центра энергосбережения Псковского государственного политехнического института. – Псков: Издательство ППИ, 2009.
4. Пресс-служба МРСК Северо-Запада Потребление энергии в Псковской области в 2009 // Нефть России: Новости ТЭК. – Режим доступа : www.oilru.com.
5. Цымбал Сергей, Коптелов Андрей. Интеллектуальные технологии в электроэнергетике / ЭнергоРынок, №4, 2010.

В.Ю. ГУБАРЕВ

«ПРЫЖОК» ПРИ РЕШЕНИИ NP-ПОЛНОЙ ЗАДАЧИ

В работе рассматриваются различные подходы по оптимизации поиска решений NP-полных задач, в частности, нахождение маршрута шахматного коня, проходящего через все поля доски по одному разу. Проведён анализ задачи подсчёта количества различных маршрутов шахматного коня, проходящего через все поля доски по одному разу, показаны некоторые вычислительные сложности и предложен метод их снижения, данный метод назван «прыжком».

В теории алгоритмов NP-полные (NPC) задачи с вычислительной точки зрения являются самыми сложными из класса NP. Для их решения не найдено ни одного алгоритма, вычислительная сложность которого росла бы пропорционально росту полиномиальной функции от объёма входных данных (или их размерности).

Кроме того, если бы нашёлся хотя бы один такой алгоритм, то с помощью его можно было бы решить все задачи, относящиеся к классу NP, так как работы С. А. Кука[1] и, позже, Р. М. Карпа[2] показали, что любую задачу класса NP можно за «полиномиальное время» свести к любой NP-полной задачи.

В связи с вышеизложенными фактами для простоты восприятия информации была выбрана NP-полная задача, которой занимался сам Леонард Эйлер[3], нахождение маршрута шахматного коня, проходящего через все поля доски по одному разу.

---

[1] Стивен Артур Кук (англ. StephenArthurCook, 14 декабря 1939 года, Буффало, США) – американский учёный в области теории вычислительных систем. Знаменит своей работой над теорией сложности вычислений, лауреат премии Тьюринга. В своей работе «TheComplexityofTheoremProvingProcedures» Кук доказал, что задача выполнимости булевых формул является NP-полной. Тем самым он поднял вопрос о равенстве классов сложности P и NP, один из сложнейших вопросов теории вычислительных систем, на который до сих пор нет ответа.

[2] Ричард Мэннинг Карп (англ. RichardManningKarp, 3 января 1935 года, Бостон, США) – американский учёный в области теории вычислительных систем, в 1985 году получил Премия Тьюринга «за его продолжительный вклад в теорию алгоритмов, в том числе за разработку эффективных алгоритмов для потоков на сетях и других комбинаторных оптимизационных задач, сопоставление вычислений полиномиальной сложности с интуитивным понятием эффективности, и, самое главное, за вклад в теорию NP-полноты».

[3] Леона́рдЭ́йлер (нем. LeonhardEuler; 4 (15) апреля 1707, Базель, Швейцария – 7 (18) сентября 1783, Санкт-Петербург, Российская империя) – российский, немецкий и швейцарский математик, внёсший значительный вклад в развитие математики, а также механики, физики, астрономии и ряда прикладных наук. Леонард Эйлер посвятил задаче «О нахождении маршрута шахматного коня, проходящего через все поля доски по одному разу» большую работу «Решение одного любопытного вопроса, который, кажется, не подчиняется никакому исследованию» (датируется 26 апреля 1757 года). В письме к Гольдбаху он сообщал:

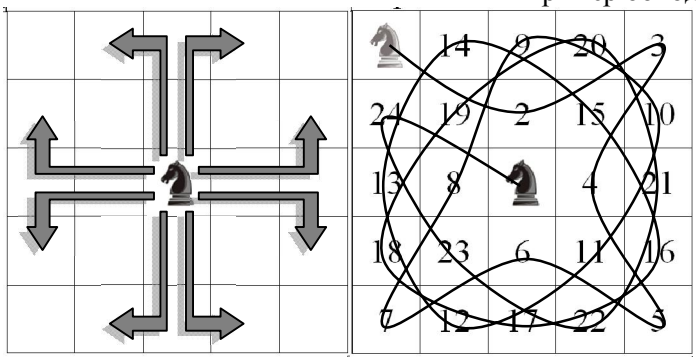
В терминах теории графов каждый маршрут коня, проходящий через все поля шахматной доски, соответствует гамильтонову пути в графе, вершинами которого являются поля доски, и два поля соединены ребром, если с одного можно попасть на другое за один ход шахматного коня.

К настоящему времени так же известно (в соответствии с различными источниками, в том числе и сети Интернет), что количество всех замкнутых маршрутов коня (гамильтоновых циклов) на поле 8×8 без учёта направления обхода равно 13 267 364 410 532 (количество замкнутых маршрутов с учётом направления в два раза больше). В то же время задача подсчёта всех возможных незамкнутых маршрутов значительно сложнее и не решена до сих пор. Известно, что количество незамкнутых маршрутов не превышает числа сочетаний:

$$\binom{168}{63} = 1{,}2 \cdot 10^{47}$$ стр. 22 [1]

Существует множество различных эвристических способов нахождения «гамильтонова пути коня». Один из самых простых и эффективных способов называется «правилом Варнсдорфа[4]» и, по сути, является разновидностью жадного» алгоритма.

Пример обхода конём поля 5×5



Этот алгоритм линейный по времени и не требует выделения большого количества памяти. Но всё же является эвристическим и на большом поле шахматный конь может «заблудиться».

Кроме того, в начале статьи поставлена несколько иная задача и для её решения нет (по крайней мере, на данный момент) асимптотически более эффективных методов, чем полный перебор.

И всё-таки можно выделить несколько способов решений, предлагаемых в различных бумажных и электронных источниках информации, для решения поставленной задачи (подсчёт количества различных путей коня с посещением каждого поля прямоугольной шахматной доски (размером N на M) точно по одному разу).

1. Полный перебор[5].

Данный способ самый медленный. Пусть числами от 1 до $(N \cdot M)$ будут нумероваться позиции коня в порядке обхода полей шахматной доски. Тогда существует $(N \cdot M)!$ (факториал) различных заполнений клеток доски этими числами. Из всего этого

«…Воспоминание о предложенной когда-то мне задаче послужило для меня недавно поводом к некоторым тонким изысканиям, в которых обыкновенный анализ, как кажется, не имеет никакого применения… Я нашел, наконец, ясный способ находить сколько угодно решений (число их, однако, не бесконечно), не делая проб».

[4] При обходе доски конь перемещается на то поле, с которого можно пойти на минимальное число ещё не пройденных полей. Если таких полей несколько, то можно пойти на любое из них.

[5] Полный перебор (или метод «грубой силы» от англ. bruteforce) – метод решения задачи путем перебора всех возможных вариантов. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий.

Любая задача из класса NP может быть решена полным перебором. При этом, даже если вычисление целевой функции от каждого конкретного возможного решения задачи может быть осуществлена за полиномиальное время, в зависимости от количества всех возможных решений, полный перебор может потребовать экспоненциального времени работы.

множества остаётся лишь выбрать, те, которые являются «гамильтоновыми путями коня» и сосчитать их.

Для оценки сложности приведём таблицу факториалов $(N \cdot M)!$

Количество строк (N) \ Количество столбцов (M)	1	2	3	4	5	6	7	8
1	1	2	6	24	120	$7{,}2\cdot10^{2}$	$5{,}0\cdot10^{3}$	$4{,}0\cdot10^{4}$
2	2	24	$7{,}2\cdot10^{2}$	$4{,}0\cdot10^{4}$	$3{,}6\cdot10^{6}$	$4{,}8\cdot10^{8}$	$8{,}7\cdot10^{8}$	$2{,}1\cdot10^{13}$
3	6	$7{,}2\cdot10^{2}$	$3{,}6\cdot10^{5}$	$4{,}8\cdot10^{8}$	$1{,}3\cdot10^{12}$	$6{,}4\cdot10^{15}$	$5{,}1\cdot10^{19}$	$6{,}2\cdot10^{23}$
4	24	$4{,}0\cdot10^{4}$	$4{,}8\cdot10^{8}$	$2{,}1\cdot10^{13}$	$2{,}4\cdot10^{18}$	$6{,}2\cdot10^{23}$	$3{,}1\cdot10^{29}$	$2{,}6\cdot10^{35}$
5	120	$3{,}6\cdot10^{6}$	$1{,}3\cdot10^{12}$	$2{,}4\cdot10^{18}$	$1{,}6\cdot10^{25}$	$2{,}7\cdot10^{32}$	$1{,}0\cdot10^{40}$	$8{,}2\cdot10^{47}$
6	$7{,}2\cdot10^{2}$	$4{,}7\cdot10^{8}$	$6{,}4\cdot10^{15}$	$6{,}2\cdot10^{23}$	$2{,}7\cdot10^{32}$	$3{,}7\cdot10^{41}$	$1{,}4\cdot10^{51}$	$1{,}2\cdot10^{61}$
7	$5{,}0\cdot10^{3}$	$8{,}7\cdot10^{10}$	$5{,}1\cdot10^{19}$	$3{,}1\cdot10^{29}$	$1{,}0\cdot10^{40}$	$1{,}4\cdot10^{51}$	$6{,}1\cdot10^{62}$	$7{,}1\cdot10^{74}$
8	$4{,}0\cdot10^{4}$	$2{,}1\cdot10^{13}$	$6{,}2\cdot10^{23}$	$2{,}6\cdot10^{35}$	$8{,}2\cdot10^{47}$	$1{,}2\cdot10^{61}$	$7{,}1\cdot10^{74}$	$1{,}3\cdot10^{89}$

В году приблизительно $3\cdot10^{7}$ секунд, будем считать, что мы можем перебирать за дну секунду 10^{8} вариантов, то для поля 5×5 нам потребуется более 10^{9} лет. Полный перебор почти никогда не используется.

2. Поиск с возвратом[6].

Применительно к разбираемой задаче данный метод будет действовать следующим образом. Вначале «конь» помещается на одно из полей, совершается допустимый ход, затем следующий и так далее, до тех пор, пока не будет возможности сделать ход по правилам хода шахматного коня. Далее оцениваем, сколько ходов было сделано (все ли клетки шахматной доски были пройдены). Если были посещены все клетки, то увеличиваем количество найденных вариантов обхода на единицу, в противном случае этого не делаем. Далее возвращаемся в предыдущее состояние, то которое было до последнего хода, и пытаемся сделать другой ход. Когда все варианты исчерпаны, возвращаемся ещё на одну позицию назад и процесс повторяется аналогично только что описанному. Программисты для реализации такого подхода обычно используют рекурсию[7].

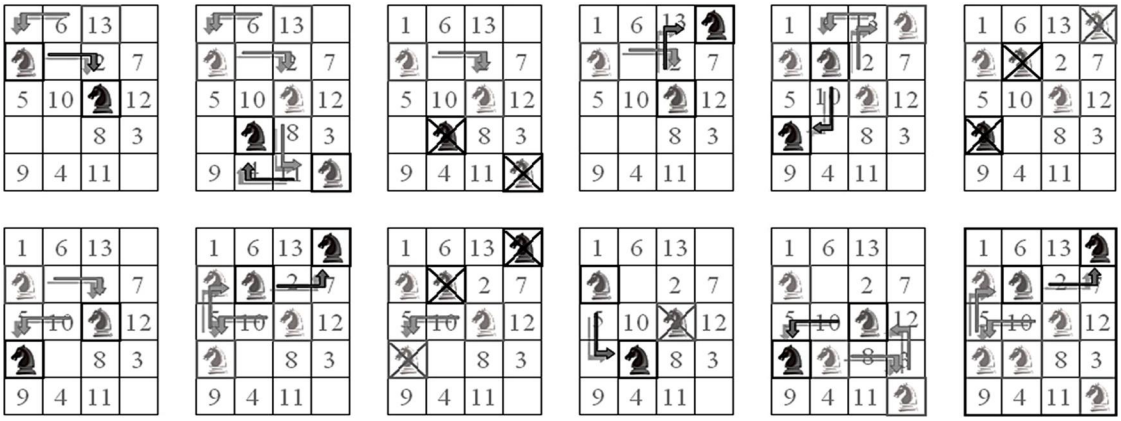
На следующем рисунке читателю предлагается проследить работу метода «поиска с возвратом» на примере обхода шахматным конём доски размером 5×4. Пошагово процесс показан начиная с 14 хода, а предыдущие посещения клеток пронумерованы в хронологическом порядке.

[6] Поиск с возвратом (англ. backtracking) – общий метод нахождения решений задачи, в которой требуется полный перебор всех возможных вариантов в некотором множестве A. Как правило, данный метод позволяет решать задачи, в которых ставятся вопросы типа: «Перечислите все возможные варианты …», «Сколько существует способов …», «Есть ли способ …», «Существует ли объект…» и т. п.

Решение задачи методом поиска с возвратом сводится к последовательному расширению частичного решения. Если на очередном шаге такое расширение провести не удается, то возвращаются к более короткому частичному решению и продолжают поиск дальше. Данный алгоритм позволяет найти все решения поставленной задачи, если они существуют.

[7] Рекурсия – вызов функции (процедуры) из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная рекурсия), например, функция A вызывает функцию B, а функция B – функцию A. Количество вложенных вызовов функции или процедуры называется глубиной рекурсии.

Пример работы алгоритма на шахматной доске 5×4



Для оценки сложности (объёма вычислений) приведём таблицу с количеством времени выполнения, затраченного на поиск всех решений. Поиск будем запускать только из одного углового поля. В качестве вычислительного средства использовался ПК с процессором работающем на частоте 1,8 ГГц.

Количества найденных обходов и время выполнения программы

Кол-во строк (N) \ Количество столбцов (M)	3	4	5	6	7	8
3	0 0 сек	2 0 сек	0 0 сек	0 0 сек	8 0,078 сек	82 0,609 сек
4	2 0 сек	0 0 сек	32 0 сек	220 0,063 сек	1682 1,4 сек	7630 27 сек
5	0 0 сек	32 0 сек	304 0,172 сек	4542 10 сек	187432 626 сек	
6	0 0 сек	220 0,063 сек	4542 10 сек	524486 1926 сек		

Дополнительно укажем, при работе алгоритма на поле 5×5 компьютер произвёл шахматным конём 1 735 079 ходов, при размере 6×5 – 98 742 288, а при 6x6 это число уже выросло до 19 431 824 959.

3. Метод ветвей и границ[8].

Данный метод в рассматриваемой задаче является вырожденным, так как условие отсева подмножеств, заранее известно, и не изменяется при работе алгоритма. Применительно к рассматриваемой задаче, обходя дерево решений, часто возникает ситуация разделения шахматного поля на две и более части, то есть не посещённые поля не образуют связанную область. В теории графов говорят, что связный граф распался на компоненты связности. Тогда нет смысла продолжать перебор, так как обойти все оставшиеся поля мы не сможем. Для определения связности графа требуется дополнительное время, но оно окупается (см. таблицу).

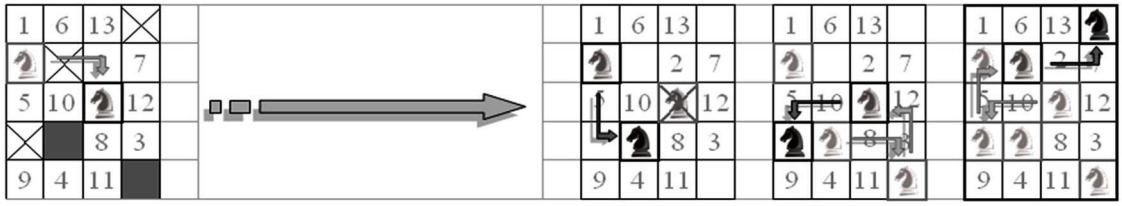
[8] Метод ветвей и границ (англ. branchandbound) – общий алгоритмический метод для нахождения оптимальных решений различных задач оптимизации, особенно дискретной и комбинаторной оптимизации. По существу, метод является вариацией полного перебора (перебора с возвратом) с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений. Для ускорения метода стараются вычисления организовать таким образом, чтобы как можно раньше выявлять заведомо неподходящие варианты. Зачастую это позволяет значительно уменьшить время нахождения решения.

Количество найденных обходов и время выполнения программы

Кол-во строк (N) \ Количество столбцов (M)	3	4	5	6	7	8
3	0 0 сек	2 0 сек	0 0 сек	0 0 сек	8 0 сек	82 0 сек
4	2 0 сек	0 0 сек	32 0 сек	220 0,032 сек	1682 0,531 сек	7630 6 сек
5	0 0 сек	32 0 сек	304 0,063 сек	4542 2,765 сек	187432 117 сек	
6	0 0 сек	220 0,032 сек	4542 2,765 сек	524486 402 сек		

Для сравнения, при работе алгоритма на поле 5×5 компьютер произвёл шахматным конём 79 891 ход, при размере 6×5 – 2 887 189, а при 6×6 – 360 970 435.

Пример работы алгоритма на шахматной доске 5×4



Кроме указанной оптимизации следует также опробовать эффективность отсечения «по потере двусвязанности графа». Более точно, следует производить возврат к предыдущей позиции, если граф имеет точку сочленения, удаление которой приведёт к образованию более двух компонент связанности.

4. Метод «Динамического программирования»[9].

Рассмотрим, как динамическое программирование (ДП) может быть применимо для рассматриваемой нами задачи. В качестве примера рассмотрим обход на поле 5×4 на глубину 7 ходов (8 вершин).

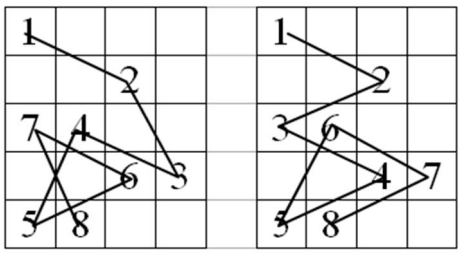
На рисунке представлено два различных обхода, но остальная часть и поле продолжения обхода идентичны, следовательно, далее следует искать обход из только одной позиции, а количество найденных решений увеличить вдвое. Покажем, насколько эффективен предложенный вариант при подсчёте количества различных маршрутов шахматного коня на примере обхода доски 6×6.

Количество позиций с *одной компонентой связанности на разной глубине дерева* решений

Глубина перебора	7	8	9	10	11	12	13	14
без ДП	1890	6224	18736	54568	145200	367778	831360	1790860
с ДП	1874	5978	17586	48642	122946	285084	581117	1084878

Глубина перебора	15	16	17	18	19	20	21	22
без ДП	3367002	6050872	9353268	13988452	17828908	22359968	23627662	25177028
с ДП	1723978	2565836	3214030	3892860	3940290	3962118	3295809	2782232

[9] Динамическое программирование – метод решения задач с оптимальной подструктурой и перекрывающимися подзадачами, который намного эффективнее, чем решение «в лоб» (bruteforce). Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи. Перекрывающиеся подзадачи в динамическом программировании означают подзадачи, которые используются для решения нескольких задач большего размера.



Заметим, что при глубине перебора на 21 ход (22 пройденных поля) получается уже девятикратный выигрыш в количестве позиций, из которых необходимо продолжать поиск решения.

Но что плохо, и об этом пишут в различной специализированной литературе, это затраты памяти. Пик вариантов был, достигнут при глубине в 20 полей, и составил 3 962 118 позиций. Так вот, если для хранения каждой позиции будет затрачено по 8 байт, то потребуется 32 МБ памяти. Вроде немного, но на поле 7×6 на глубине 17 уже требуется 320 МБ, а ещё не обошли и половины полей. То есть появляется острая проблема в другом ресурсе – памяти.

5. «Прыжок».

При исследовании задачи методом динамического программирования было обнаружено, что изменение потребности в памяти имеет вид «горки». То есть по мере увеличения глубины перебора количество необходимой памяти сначала резко растёт, затем, достигнув максимальной своей величины, почти также резко уменьшается. Этот факт наблюдается независимо от размерности задачи.

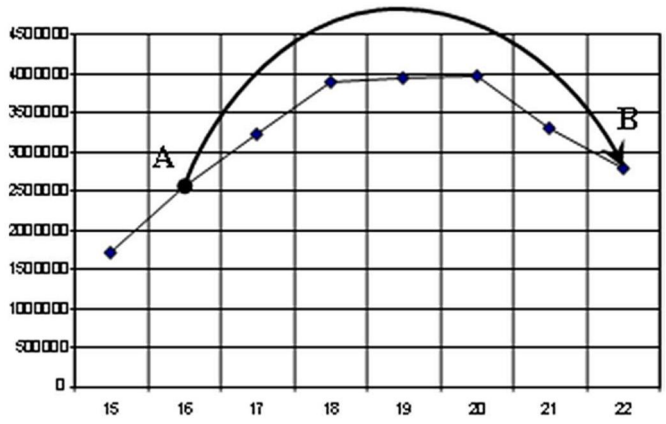
Так появилась идея из состояния А «перепрыгнуть» в состояние В. Где А – это состояние до «вершины горки», а В – после.

При таком подходе имеется возможность использовать метод ДП, до тех пор, пока есть возможность выделять необходимую память, а затем, например, перебором с возвратом, не прибегая к хранению состояний на каждом шаге углубления, «добраться» до состояния В, где опять-таки использовать ДП.

Данный метод назван «Прыжком», так как, откладывая на графике, на участке без использования ДП, количество перебранных вариантов на каждом шаге получается дуга над «горкой ДП» и, вследствие того, что на каждом моменте этого участка мы не задерживаемся, это напомнило мне прыжки через препятствия.

6. Ещё быстрее!

Даже если бы мы могли использовать динамическое программирование, как было описано выше, на всём протяжении работы алгоритма (без «прыжка»), то есть обладали бы «бесконечным» количеством оперативной памяти, то всё равно получение результата потребовало бы колоссального количества времени. Ожидаемое время работы такого алгоритма на ПК для поля 8×8 порядка 1000 лет (против 10^9 лет), а количество маршрутов не должно превысить 10^{20} вариантов. Последние оценки были вычислены из полученных данных при запуске программ на глубину 20 на поле 8×8 и предположением поведения роста числа вариантов аналогично результатам с меньшими полями.



Дальнейшее направление исследования в настоящий момент направлено на использование парадигмы «разделяй и властвуй» и использованием предпросчитанных позиций. Парадигма «разделяй и властвуй» будет использоваться совместно с алгоритмом обнаружения точек сочленения, которые и будут использоваться как разделители. Рассмотрим примеры использования парадигмы.

На рисунке слева показан вариант обхода глубиной в 11 ходов, при котором нет смысла продолжать углубляться, так как цель всё равно не будет достигнута, об этом сигнализирует выделенная серым цветом «точка сочленения» соединяющая 3-и не связанные области (несвязанные области помечены различными фигурками, наибольшая область оставлена чистой). Примечание: посещение всех трёх частей потребует как

минимум двукратного посещения точки сочленения, а это противоречит условию задачи. Под удаление также попадают позиции, при которых двусвязные области имеют более 2-х точек сочленения.

Также возможны ситуации (и их не мало), когда точки сочленения разделяют оставшуюся область на две части, в таком случае количество обходов будет равно произведению обходов каждой из подобластей в разном сочетании. Небольшие части, получаемые при разбиении, можно предпросчитать и упорядочить, в результате для таких частей асимптотически ответ будет получен за O(log A), где A – количество различных (с учётом поворотов, переносов и отражений) частей. Для частей бóльших, чем предпросчитанные, необходимо продолжить обход и разделение на части.

ЛИТЕРАТУРА

1. Шахматы и математика. Е.Я. Гик. – Библиотечка «Квант», выпуск 24. – М.: Наука, 1983.
2. Компьютер и задачи выбора. – Академия наук, серия «Кибернетика – неограниченные возможности и возможные ограничения». – М.: Наука, 1983.

И.Н. КОЗЫРЕВ, С.Ф. ГРУНИН

ЗАЩИТА СИСТЕМ БЕСПЕРЕБОЙНОГО ЭЛЕКТРОПИТАНИЯ

Приводятся требования и краткое описание устройства защиты систем бесперебойного электропитания.

Современные потребители электроэнергии характеризуются наличием электроустановок, предъявляющих различные требования к надежности электроснабжения и к показателям качества электроэнергии. Это определяет широкое применение систем бесперебойного (СБЭ) электропитания, предназначеных для автономного электроснабжения особо ответственных электроприемников в случае нарушения их электроснабжения от основных источников электроэнергии (ОИЭ). Основу СБЭ составляют источники бесперебойного питания (ИБП), системы постоянного тока и аккумуляторные батареи.

Наиболее широкое применение СБЭ нашли в системах обработки и передачи информации, для электропитания оптоволоконных линий связи. В качестве ОИЭ оптоволоконных систем передачи данных зачастую используются как электрические сети III категории, так и электрические сети железнодорожного транспорта. К числу основных особенностей указанных сетей можно отнести наличие мощных источников помех, таких как линии электропередачи, тяговые подстанции, подвижной состав с соответствующими коммутациями тягового тока, грозовые разряды, процессы переключения питающих фидеров стационарных систем электроснабжения.

Стандартом IEEEStd 1100-1999 "IEEE Recommended Practiceforpowerand Grounding Electronic Equipment", IEEE Press, 1999 установлены требования по устойчивости к воздействиям изменения сетевого напряжения. В дальнейшем рабочей группой CBEMA была получена уточненная зависимость, одобренная Советом Индустрии Информационных Технологий ITIC (InformationTechnologyIndustryCouncil). Данная зависимость получила название кривой ITIC. Данная кривая приведена на рисунке 1.

Кривая показывает зависимость между величиной отклонения напряжения и его предельно допустимой длительностью.

Для защиты входных цепей дорогостоящих ИБП от перенапряжений, возникающих в ОИЭ, предназначено устройство сетевой защиты (УСЗ), разработанное сотрудниками кафедры электроэнергетики ПГПИ. Аппаратное и программное обеспечение данного устройства реализует кривую, приведенную на рисунке 2. Данная кривая применима к любому оборудования, включающему полупроводниковые устройства. При разработке этой кривой учтены требования [1], [2], [3]. Кривая максимально приближена к