КОМПЬЮТЕРНАЯ МАТЕМАТИКА

УДК 519.8

В.В. ТУРИНСКИЙ

РАЗРАБОТКА И ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ МЕТАЭВРИСТИЧЕСКИХ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧ ПЛАНИРОВАНИЯ РАБОТЫ НЕЗАВИСИМЫХ МАШИН

Рассмотрен один класс задач теории расписаний по планированию работы независимых машин разной производительности. Предложены и реализованы программно четыре метаэвристических алгоритма решения задач данного класа. Исселедованы вопросы их эффективности на основе анализа результатов вычислительного эксперимента с использованием серии известных задач.

Введение. Рассматривается задача составления расписания обработки заданного множества заданий при наличии ряда ограничений на машинах или приборах, имеющих разную производительность [1 – 3]. Используется предложенная в [3] математическая модель, которая при соответствующей интерпретации также может описывать ряд задач размещения прямоугольников на полубесконечной ленте, состоящей из нескольких полос [4].

Цель работы состоит в разработке ряда метаэвристических алгоритмов решения задачи и исследования их практической эффективности при решении серии известных задач.

Предложенный в [3] подход позволяет перейти от рассматриваемой задачи нелинейного программирования к соответствующей задаче комбинаторной оптимизации на перестановках и дает возможность применять известные алгоритмы решения задач такого типа. Таким образом, для решения поставленной задачи в работе разработаны такие алгоритмы, как алгоритм имитационного отжига, генетический алгоритм, *H*-метод и алгоритм оптимизации муравьиными колониями (ОМК) и исследованы вопросы их эффективности на основе вычислительного эксперимента.

Разработанные алгоритмы решения. Для решения задачи разработаны и реализованы такие алгоритмы комбинаторной оптимизации: алгоритм имитационного отжига, *H*-метод, генетический алгоритм и алгоритм оптимизации муравьиными колониями. Пространство решений задачи – это пространство перестановок, компонентами которых являются номера работ. Процедура перехода от произвольной перестановки к соответствующему расписанию, используемая для расчета целевой функции, а также доказательство корректности этого перехода приведены в работе [3]. Изложим кратко вычислительные схемы разработанных алгоритмов, более детальное описание которых представлено в [5].

Алгоритм имитационного отжига. Особенностью предложенной реализации алгоритма является процедура нагревания, которая поднимает температуру до момента, пока вероятность принятия случайного решения из окрестности не будет равна заранее заданному пороговому значению. Это позволяет решать задачи с более широким диапазоном возможных значений целевой функции. В качестве окрестности была использована окрестность транспозиций элементов перестановки. Вычислительная схема алгоритма показана на рис. 1.

```
procedure АИО;
    begin
        < Создание случайного решения X >;
        T := < Процедура нагревания >;
        Record := X;
        repeat
            for i = 1 to K do
                Y := < Очередное случайное решение из окрестности N(X) >;
                p := e^(-(ЦФ(X)-ЦФ(Y))/T)
                r := < Случайное число из [0, 1] >;
                if (r<p) then
                    X :=Y;
                    if ЦФ(Y) < ЦФ (Record) then
                        Record := Y;
                    end
                else if < решение не принималось |N(X)| итераций > then
                    break
                end
            end
            T := g(T);
        until T < T_k
    end
```

РИС. 1. Алгоритм имитационного отжига

На рис. 1 использованы следующие обозначения:

ЦФ(X) – это значение целевой функции задачи для произвольного решения X;

$N(X)$ – множество соседних к X решений, которые образованы транспозицией одной пары компонент X;

$|N(X)|$ – мощность множества $N(X)$;

K – максимальное количество итераций для заданной температуры.

Процедура нагревания реализована согласно такой схемы:

1) начальная температура подбирается экспериментальным путем и обычно равна какому-либо небольшому значению, например, единице;

2) для выбранной температуры производится небольшое количество (от 50 до 100) итераций основного алгоритма, принимая каждое случайное решение из множества соседних, но при этом сохраняя вероятность принятия этого решения;

3) если средняя вероятность принятия решения меньше выбранного заранее значения, начальная температура увеличивается, и возвращаемся к п. 1). Рекомендуемые значения вероятности принятия решения лежат в интервале (0.7; 0.9).

В результате анализа результатов серии вычислений предложена формула увеличения температуры

$$T_{i+1} = (1.8 - P_i)T_i,$$

где P_i – средняя вероятность принятия решения, i – номер итерации.

Генетический алгоритм – это популярный класс эволюционных алгоритмов. Обычно, основную роль играет применения оператора рекомбинации к двум решениям, а также оператор мутации, который случайно модифицирует решения для диверсификации поиска.

В реализованном генетическом алгоритме отбор для рекомбинации, мутации и выживания происходит пропорционально приспособленности особи методом рулетки. Для рекомбинации был использован порядковый кроссинговер, а для мутации – случайная транспозиция двух компонент перестановки. Вычислительная схема генетического алгоритма показана на рис. 2, где использованы общепринятые в эволюционных алгоритмах обозначения [5].

***H*-метод.** Этот метод имеет структуру, схожую с генетическим алгоритмом, с теми отличиями, что используемые в нем процедуры можно рассматривать как обобщение процедур рекомбинации и мутации, осуществляемые независимо. Важно отметить, что *H*-метод относится к алгоритмам глобального поиска. Более подробно его вычислительная схема излагается в [6].

Алгоритм оптимизации муравьиными колониями. Основная идея алгоритма ОМК – это имитация коллективной роботы настоящих муравьев при поиске путей к пище [5]. Предложенные метаэвристики на основе ОМК могут рассматриваться как мультиагентные системы, где действия каждого агента основаны на моделировании поведения настоящего муравья.

Алгоритм основан на идее, что каждый муравей оставляет за собой след из феромона, который со временем испаряется. Из-за этого, более короткие пути имеют более интенсивный след, чем длинные. Пути, по которым прошло больше муравьев, тоже будут иметь большее количество феромона. При выборе пути муравьи руководствуются присутствующим количеством феромона и некоторой специфической для задачи эвристикой.

```
procedure ГА;
    begin
        t := 0;
        for i = 0 to m – 1 do
            X := < Создание случайного решения >;
            P^0 := P^0 ∪ X;      {P^0 – начальная популяция}
        end
        repeat
            dP := {}
            for i = 0 to m_c do
                a, b := < Отбор для рекомбинации из P^t >;
                X := Рекомбинация (a, b);
                dP := dP ∪ X;
            end
            for i = 0 to m_m do
                X := < Отбор для мутации из dP >;
                X := Мутация(X);
                dP := dP ∪ X;
            end
            P^t := P^t + dP;
            P^{t+1} := < Отбор для выживания из P^t >;
            t := t + 1;
        until < Лучшее решение не изменялось K популяций >
    end
```

РИС. 2. Генетический алгоритм

Реализация алгоритма близка к общей схеме, изложенной в [5] (рис. 3), где используются общепринятые для алгоритма ОМК обозначения. В этой реализации алгоритма используется так называемый «демон» – процедура, которая на лучшем пути, найденном одним поколением муравьев, дополнительно увеличивается количество феромона.

На рис. 3 $m, \tau_0, \Delta\tau, \rho, \alpha, \beta$ – это обозначения стандартных параметров в алгоритмах ОМК.

```
procedure ОМК;
    begin
        t := 0; τ_ij := τ_0,   i, j ∈ {0,..., n – 1};
        repeat
            P := < Новая популяция размера m >;
            for ant ∈ {0,..., m – 1} do
                i = < Случайный номер работы >
                S = {0,..., n – 1}   {S – множество номеров не назначенных работ}
```

while ($S \neq \varnothing$) **do**

j := < Номер следующей работы из S >;

{ Номер выбирается с вероятностью $\dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{j=0}^{m} \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}$ }

$S = S - \{j\}$

$\tau_{ij} := \tau_{ij} + \Delta\tau$;

$i := j$;

end

end

X := < Лучшее решение из P >;

X := Локальный Поиск (X);

for $k = 0$ **to** $n - 1$ **do**

$i := X_k$;

$j := X_{k+1}$;

$\tau_{ij} := \tau_{ij} + \Delta\tau$;

end

for $i = 0$ **to** $n - 1$ **do**

for $j = 0$ **to** $n - 1$ **do**

$\tau_{ij} := \tau_{ij} * (1 - \rho)$

until < Лучшее решение не изменялось K популяций >

end

РИС. 3. Алгоритм оптимизации муравьиными колониями

Результаты вычислительного эксперимента. В качестве исходных данных были взяты задачи из известной библиотеки [7], сгенерированные с помощью процедур, изложенных в [8]. Они генерировались с учетом того, какие условия чаще всего встречаются в литературе и использовались для проведения вычислительных экспериментов в нескольких опубликованных исследованиях. Среди условий есть как типовые для этого класса задач условия с временем выполнения работ, равномерно распределенном в интервалах $U(1,100)$ и $U(10,100)$, так и дополнительные условия с интервалами $U(100,200)$, $U(100,120)$ и $U(1000,1100)$. Для большей их части известно оптимальное решение, найденное с помощью системы IBM-ILOG CPLEX 11.0 на кластере персональных компьютеров [9]. Расчеты производились на ПК Intel Core i7-930 3.36 GHz, оперативная память – 6 GB.

Из 1400 задач, приведенных в [7], выбрано 140 задач с разными параметрами генерации. Учитывая что каждую задачу нужно решать четырьмя алгоритмами, делая по три перезапуска, для всех алгоритмов введено ограничение по времени на решение одной задачи равное пяти минутам.

Решения, полученные разработанными алгоритмами, сравнивались с известными оптимумами решаемых задач. В таблице приведена средняя процентная ошибка решений задач, рассчитанная по формуле:

$$q = \frac{f - f^*}{f^*} \cdot 100\%,$$

где f – значение ЦФ, найденное соответствующим алгоритмом, f^* – значение ЦФ, найденное на кластере с помощью системы CPLEX. Для каждой задачи было сделано три перезапуска и выбрано лучшее решение. Учитывая большое количество задач, в таблице приведены только результаты решения выбранного подмножества задач.

ТАБЛИЦА. Средняя процентная ошибка решений задач

Задача	Средняя процентная ошибка			
	АИО	Н	ГА	ОМК
u_1_100/111.txt	0.00	4.27	5.98	5.13
u_1_100/531.txt	14.52	22.58	20.97	19.35
u_1_100/1051.txt	23.53	29.41	27.45	25.49
u_10_100/111.txt	1.52	6.06	7.07	9.09
u_10_100/531.txt	13.21	16.98	15.09	16.51
u_10_100/1051.txt	15.86	17.62	16.30	16.74
u_100_120/111.txt	0.20	1.08	1.38	2.07
u_100_120/531.txt	1.47	2.17	1.94	1.88
u_100_120/1051.txt	2.99	5.04	3.54	3.69
u_100_200/111.txt	0.54	3.33	6.3	6.75
u_100_200/531.txt	6.33	9.21	7.36	8.46
u_100_200/1051.txt	8.01	9.23	8.74	8.94
u_1000_1100/111.txt	0.00	0.49	0.60	0.83
u_1000_1100/531.txt	0.62	0.98	0.89	0.92
u_1000_1100/1051.txt	1.85	2.52	1.74	2.07
maq_corre/111.txt	2.03	4.06	5.42	5.42
maq_corre/531.txt	9.33	13.16	11.0	12.2
maq_corre/1051.txt	7.74	9.88	8.93	9.17

job_corre/111.txt	2.38	5.74	5.74	4.55
job_corre/531.txt	5.62	7.99	6.19	4.84
job_corre/1051.txt	4.98	7.13	5.96	4.00
Среднее значение	**5.84**	**8.52**	**8.03**	**8.0**

Для алгоритма имитационного отжига использовалось геометрическое температурное расписание, определяемое выражением $g(T) = 0.97 \cdot T$, а конечная температура $T_k = 0.01$.

В генетическом алгоритме использовались следующие параметры: размер популяции – 3000; количество пар особей для скрещивания – 300; количество операций мутаций – 50.

Для *H*-метода выбраны следующие значения параметров: размер популяции – 2000; количество пар для вариации – 200; количество операций мутации – 50.

Алгоритм оптимизации муравьиными колониями запускался с использованием таких значений параметров: размер популяции муравьев $m = 1000$; $\tau_0 = 0.001$; $\Delta\tau = 0.01$; $\rho = 0.3$; $\alpha = 1$; $\beta = 3$.

Как можно заметить из таблицы, в задачах с меньшим разбросом длительности работ алгоритмы дают результаты, сопоставимые с известными решениями. При этом задачи большей размерности с большим разбросом длительностей решаются немного хуже. Как отмечается в [8, 9], реальные задачи в большинстве своем больше схожи по своим характеристикам к задачам типа *U*(100, 200), *U*(100, 120) и *U*(1000, 1100), чем к задачам *U*(1, 100) и *U*(10, 100).

Алгоритм имитации отжига был единственным алгоритмом, который завершался раньше отведенного ограничения по времени. Из этого можно сделать вывод о том, что три других алгоритма имеют потенциал к нахождению лучших решений при выделении им большего количества времени.

Заключение. В работе рассмотрен важный класс задач теории расписаний по планированию работы независимых машин разной производительности. С помощью подхода, предложенного в [3], который позволяет перейти от задачи нелинейного программирования к соответствующей задаче комбинаторной оптимизации на перестановках без потери глобального решения, разработаны четыре метаэвристических алгоритма.

На основе вычислительного эксперимента с использованием задач из известной библиотеки продемонстрирована эффективность работы реализованных алгоритмов в условиях весьма ограниченного лимита времени на компьютере небольшой производительности.

При этом представляет интерес исследование точности популяционных алгоритмов при увеличивающихся лимитах времени, выделяемого на решение задач.

В.В. Турінський

РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕТАЕВРИСТИЧНИХ АЛГОРИТМІВ РОЗВ'ЯЗАННЯ ЗАДАЧ ПЛАНУВАННЯ РОБОТИ НЕЗАЛЕЖНИХ МАШИН

Розглянуто один клас задач теорії розкладів по плануванню роботи незалежних машин різної продуктивності. Запропоновано і розроблено програмно чотири метаевристичних алгоритми розв'язання задач даного класу. Досліджені питання їх ефективності на основі результатів обчислювального експерименту з використанням серії відомих задач.
V.V. Turinskyi

DEVELOPMENT AND IMPLEMENTATION OF METAHEURISTIC ALGORITHMS FOR SOLVING INDEPENDENT MACHINE SCHEDULING PROBLEMS

The paper concerns with research of a class of scheduling problems for parallel machines with different productivity. Four metaheuristic algorithms for solving problems of this class are proposed and implemented. Performance of the proposed algorithms is analyzed using a benchmark of known instances.

1. *McNaughton R.* Scheduling with deadlines and loss functions. Management Science. – 1959. – **6**, N 1. – P. 1 – 12.
2. *Fanjul-Peyro L. and Ruiz R.* Iterated greedy local search methods for unrelated parallel machine scheduling. European Journal of Operational Research. – 2010. – **207**, N 1. – P. 55 – 69.
3. *Гуляницкий Л.Ф., Туринский В.В.* Математическая модель одного класса задач планирования работы независимых машин // Компьютерная математика. – 2014. – № 1. – С. 113 – 118.
4. *Сергиенко И.В., Гуляницкий Л.Ф., Малышко С.А.* О решении задач размещения одного класса // Экономика и математические методы. – 1989. – № 3. – С. 560 – 564.
5. *Talbi E.-G.* Metaheuristics: From Design to Implementation // John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.
6. *Гуляницкий Л.Ф., Сергиенко И.В.* Метаэвристический метод деформируемого многогранника в комбинаторной оптимизации // Кибернетика и системный анализ. – 2007. – № 6. – С. 70 – 79.
7. *Sistemas* de Optimización Aplicada SOA [Электронный ресурс] // Режим доступа: http://soa.iti.es
8. *Woclaw A.* Scheduling Unrelated Parallel Machines. Algorithms, Complexity, and Performance // PhD thesis, Fakultat fur Elektrotechnik, Informatik und Mathematik der Universitat Paderborn, Deutschland, 2006.
9. *Fanjul-Peyro Luis, and Rubén Ruiz* Scheduling unrelated parallel machines with optional machines and jobs selection // Computers & Operations Research. – 2012. – **39**, N 7. – P. 1745 – 1753.

Получено 20.01.2015

Об авторе:

Туринский Виталий Викторович,
аспирант Института кибернетики имени В.М. Глушкова НАН Украины.
E-mail: *vitaly.turinsky@gmail.com*