РОЗДІЛ 11. МАТЕМАТИЧНІ МЕТОДИ, МОДЕЛІ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ЕКОНОМІЦІ

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ О РАСПИСАНИИ С ДОПОЛНИТЕЛЬНЫМИ УСЛОВИЯМИ В СРЕДЕ MATLAB

SOFTWARE IMPLEMENTATION OF THE GENETIC ALGORITHM APPLYING FOR SCHEDULING PROBLEM WITH ADDITIONAL CONDITIONS IN MATLAB

УДК 338.24.01

Ковалева Е.А.
к.т.н., доцент кафедры высшей математики и экономико-математических методов Харьковский национальный экономический университет им. С. Кузнеца
Мисюра Е.Ю.
к.т.н., доцент кафедры высшей математики и экономико-математических методов Харьковский национальный экономический университет им. С. Кузнеца

В статье сформулирована реальная экономическая задача теории расписаний, которая решается с использованием генетического алгоритма. Для численного решения задачи использовались программная среда Matlab, встроенная функция ga-GeneticAlgorithm и пользовательский программный код, реализованный для трех основных аспектов генетического алгоритма, а именно скрещивания, селекции и мутации.
Ключевые слова: *задача о расписании, матроид, генетический алгоритм, программная среда Matlab.*

У статті сформульовано реальну економічну задачу теорії розкладів, яка вирішується з використанням генетичного алгоритму. Для чисельного вирішення задачі використовувалися програмне середовище Matlab, вбудована функція ga-GeneticAlgorithm і призначений для користувача програмний код, реалізований для трьох основних аспектів генетичного алгоритму, а саме схрещування, селекції та мутації.
Ключові слова: *задача про розклад, матроїд, генетичний алгоритм, програмне середовище Matlab.*

The article formulates the real economic problem of scheduling theory, which is solved using the genetic algorithm. To solve the problem numerically, we used the Matlab software environment, the built-in ga-GeneticAlgorithm function and the user program code implemented for the three main aspects of the genetic algorithm: crossing, selection and mutations.
Key words: *scheduling problem, matroid, genetic algorithm, Matlab software.*

Постановка проблемы. Задачи о расписании входят в широкий класс задач дискретной математики и представляют собой в общем виде проблему упорядочения. В работе [1] подробно описана постановка таких задач, авторы статьи приводят лишь общую формулировку: для заданных множеств работ $J = \{J_1, J_2, ..., J_n\}$ и машин (приборов) $M = \{M_1, M_2, ..., M_m\}$ ставится задача дискретной оптимизации о построении расписания, которое минимизирует время выполнения работ, стоимость и т. п.

Задачи о расписании можно разделить на две группы [1; 2]:

– задачи с прерываниями (в любой момент обслуживание требования на машине может быть прервано (с возможностью завершения позже на той же или другой машине) ради обслуживания другого требования);

– задачи без прерываний (каждое требование на машине обслуживается от начала до конца без прерываний).

В работе приводится реальная экономическая задача теории расписаний, относящаяся к задачам с прерываниями и посвященная созданию расписания рабочих смен одного из предприятий холдинга «Хлебные инвестиции». Так, в работе [3] сказано, что для повышения производственной мощности по выпуску хлеба была установлена новая батонная линия мощностью 30 тонн в сутки, запущено новое ротационное отделение мощностью 12 тонн в сутки для производства формовых сортов хлеба. Эти мероприятия позволят предприятию увеличить объем реализации свежего хлеба для обеспечения населения г. Киева и Киевской области, с одной стороны, а увеличение рабочих мест на предприятии, естественно, ставит задачу об экстренном создании расписания для новых рабочих, с другой стороны.

Для различных классов задач теории расписаний существует достаточно широкий набор алгоритмов и методов решения последних [1; 2], особый интерес среди которых представляют генетические алгоритмы (ГА). Известно, что существует ограниченная область применения данных алгоритмов, а именно задача должна быть носителем матроида. Используя теорему Радо-Эдмондса [4], в статье авторы приводят доказательство выполнения всех условий существования матроида в поставленной перед ними задаче, что делает оправданным применение ГА в данном конкретном случае.

Анализ последних исследований и публикаций. В отечественной литературе этот метод недостаточно освещается, наблюдается недостаток информации и сведений о его функционировании в контексте задач. Для написания статьи проведена работа с современной зарубежной

литературой, что позволило представить новые сведения о механизме работы данного алгоритма, рассмотреть классы задач успешного применения ГА и выявить основные проблемные зоны. Так, применение ГА в задачах теории расписаний подробно изложено в работах [5; 6], однако авторы абсолютно не приводят никаких средств компьютерной математики для численного решения последних. В работе [7] автор приводит численный эксперимент, как и в работе [8], однако их задачи отличаются от задачи статьи, являясь задачами без прерываний. В работах [9; 10] рассматриваются задачи с прерываниями, однако сами статьи носят лишь обзорных характер, авторы упоминают проблемы медленной и преждевременной сходимости к оптимальному решению, но не приводят решение этой проблемы.

Работа [11] является ярким примером использования встроенных функций ГА среды Matlab для решения технической задачи, относящейся к задачам теории расписаний. Уникальность функций скрещивания, селекции и мутации не наблюдается, проблемные зоны ГА остаются нерешенными. В то же время в работе [12] речь идет о новых подходах к мутации и скрещиванию, однако эти функции автор применяет к теории игр и не указывает никакую другую область применения его функций.

Постановка задания. После проведенного анализа литературы авторы статьи ставят перед собой следующую комплексную задачу: на примере реальной экономической задачи с дополнительными ограничениями теории расписаний одного из предприятий холдинга «Хлебные инвестиции» разработать в программной среде Matlab уникальные функции, моделирующие основные этапы ГА, а именно скрещивание, селекция и мутация; провести численный эксперимент с целью получения оптимального по количеству сотрудников временного расписания.

Изложение основного материала исследования. Задача формулируется следующим образом: на предприятии существуют три смены, а именно дневная, ночная и вечерняя смены (табл. 1). Рабочий режим предполагает производство хлеба круглосуточно (семь дней в неделю). Требуется, чтобы в каждой из рабочих смен каждый день присутствовало определенное количество работников (не меньше указанного числа) (табл. 2). В табл. 1 приведены дополнительные условия, а именно человек не может работать более одной смены в день, расписание является фиксированным, также фиксированной является смена в течение четырех дней.

Требуется для указанных в табл. 1, 2 требований для каждой из смен минимизировать затраты на работников (минимизировать общее их количество), при этом учесть дополнительные условия.

Таблица 1

Дополнительные ограничения задачи

Смены завода	Дополнительные условия
Ночная смена (00:00-08:00)	1 смена/день; фиксированное расписание; смена фиксирована в течение 4-х дней
Дневная смена (08:00-16:00)	
Вечерняя смена (16:00-00:00)	

Таблица 2

Количество работников смены (не менее указанного числа в таблице)

	Пн	Вт	Ср	Чт	Пт	Сб	Вс
Ночь	5	3	2	4	3	2	2
День	7	8	9	5	7	2	5
Вечер	9	10	10	7	11	2	2

Для последующего моделирования была проведена формализация данной задачи, описанная соотношениями (1-3).

Пусть вектор $X_j, \left(j = \overline{1, 21}\right)$ – количество работников, заступающих в текущую смену на ближайшие 4 смены. Тогда задачу можно представить таким образом:

минимизировать функцію

$$F = \sum_j X_j \to \min \qquad (1)$$

при следующих ограничениях*:

$$\begin{cases} x_1 & & & & +x_5 & +x_6 & +x_7 & \geq & 5 \\ x_1 & +x_2 & & & & +x_6 & +x_7 & \geq & 3 \\ x_1 & +x_2 & +x_3 & & & & +x_7 & \geq & 2 \\ x_1 & +x_2 & +x_3 & +x_4 & & & & \geq & 4 \\ & +x_2 & +x_3 & +x_4 & +x_5 & & & \geq & 3 \\ & & +x_3 & +x_4 & +x_5 & +x_6 & & \geq & 2 \\ & & & +x_4 & +x_5 & +x_6 & +x_7 & \geq & 2 \end{cases} \qquad (2)$$

или в матричной форме:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \geq \begin{pmatrix} 5 \\ 3 \\ 2 \\ 4 \\ 3 \\ 2 \\ 2 \end{pmatrix} \qquad (3)$$

** в работе авторы приводят математическую модель задачи для ночной смены, для остальных двух формализация задачи осуществляется аналогично*

Для решения поставленной задачи целесообразно применить ГА, исходя из следующих соображений. Рассмотрим следующий матроид: пусть носителем будет множество заданий, а независимыми множествами – успешно выполненные задания. Весом каждой заявки пусть будет его стоимость. Проверим, является ли данная пара матроидом (табл. 3).

Исходя из рассуждений, приведенных в табл. 3, можем сделать вывод, что авторы статьи аргументировано применяют ГА к численному решению поставленной задачи.

Для проведения численного эксперимента авторы статьи выбрали программную среду Matlab, исходя из следующих соображений. Во-первых, Matlab как пакет прикладных программ содержит встроенные функции, использующие ГА (надстройка Optimization Tool, функция ga-GeneticAlgorithm) с последующей визуализацией данных. Во-вторых, Matlab – это одноименный язык программирования, который используется авторами статьи для написания дополнительных программных продуктов с последующей их инсталляцией в надстройку Optimization Tool. Наконец, авторы статьи работают со структурами данных, основанными на матрицах, что делает Matlab просто незаменимым программным продуктом.

Механизм работы с генетическими алгоритмами в среде Matlab реализован двумя способами:

1) вызов функции генетических алгоритмов;

2) использование комплекта Optimization Tool.

Оба способа поставляются в числе стандартного набора функций и модулей Matlab. На взгляд авторов, намного более удобным и наглядным является второй способ работы с генетическими алгоритмами в Matlab, связанный с использованием модуля Optimization Tool. Его и рассмотрим подробнее.

Для запуска утилиты Optimization Tool следует в командной строке Matlab выполнить команду optimtool. После этого запустится пакет генетических алгоритмов, а на экране появится основное окно утилиты (рис. 1).

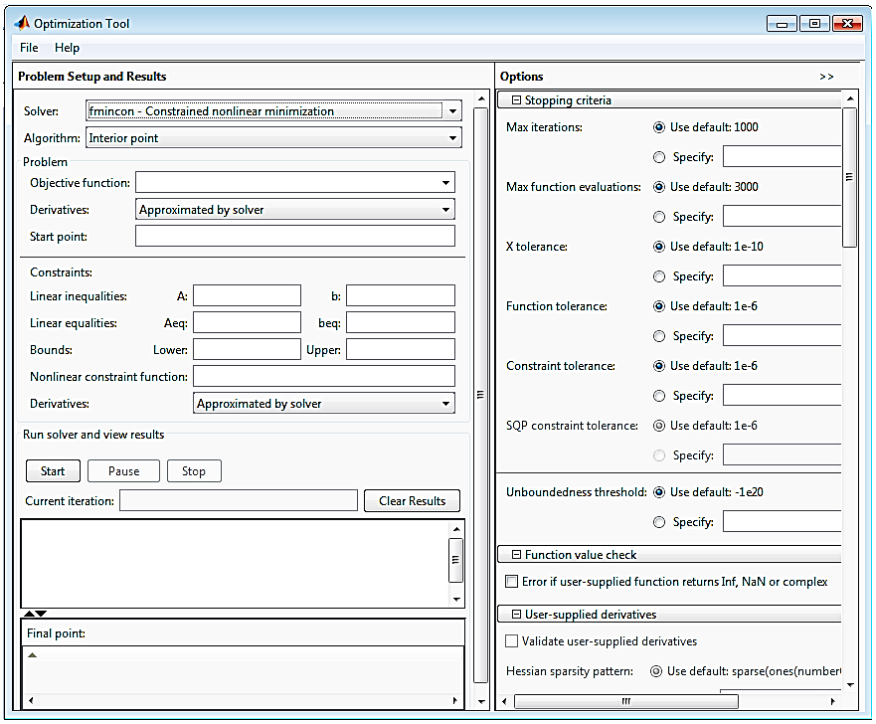


Рис. 1. Утилита Optimization Tool

Таблица 3

Применение теоремы Радо-Эдмондса к реальной задачи расписания

Постулаты теоремы Радо-Эдмондса	Применение теоремы Радо-Эдмондса к задаче, описанной соотношениями (1-3)
1. $\varnothing \in I$. Множество *I* непустое. Даже если исходное множество *X* было пустым (*X* = ∅), то *I* будет состоять из одного элемента – множества, содержащего пустое. *I* = {{∅}}.	Первое свойство, очевидно, выполняется: пустое множество выполненных заданий входит в наше множество.
2. $A \subset B, B \in I \Rightarrow A \in I$. Любое подмножество любого элемента множества *I* также будет элементом этого множества.	Второе множество тоже выполняется. Почему это так? Давайте отсортируем успешно выполненные задания в порядке увеличения крайних сроков. В таком порядке они все равно будут успешно выполненными, именно в таком порядке очевидно, что любое подмножество успешно выполненных заданий будет успешно выполнено.
3. $\lvert A\rvert < \lvert B\rvert \Rightarrow \exists x \in B \setminus A, A \cup \{x\} \in I$. Если множества *A* и *B* принадлежат множеству *I*, а также известно, что размер *A* меньше *B*, то существует какой-нибудь элемент *x* из *B*, не принадлежащий *A*, что объединение *x* и *A* будет принадлежать множеству *I*. Это свойство является не совсем тривиальным, но чаще всего наиважнейшим из всех остальных.	Третье свойство хоть и не очевидно, но выполняется. Пусть у нас есть два множества успешно выполненных заданий *A* и *B*, причем известно, что $\lvert A\rvert < \lvert B\rvert$. Стандартно отсортируем задания в порядке увеличения крайних сроков в обоих множествах. Возьмем задание из *B*, которого нет в *A*, и попробуем добавить его к множеству *A*. Это у нас получится, ведь если бы в *A* не было пробела, то данное задание должно было присутствовать.

В поле Fitness function указывается оптимизируемая функция в виде @fitnessfun, где fitnessfun.m – название М-файла, в котором предварительно следует описать оптимизируемую функцию (табл. 4). В панели Constraints можно задать ограничения или ограничивающую нелинейную функцию. В поле Linear inequalities задается линейное ограничение неравенством ($Ax \le b$), в поле Linear equalities данной панели задаются линейные ограничения равенством ($Ax = b$).

В обоих случаях *A* – некоторая матрица, *b* – вектор.

В поле Bounds в векторном виде задаются нижнее и верхнее ограничения переменных, а в поле Nonlinear constraint function можно задать произвольную нелинейную функцию ограничений.

Из рис. 1 видно, что основными настраиваемыми параметрами в Optimization Tool Options являются:

– популяция (вкладка Population);

– масштабирование (вкладка Fitness Scaling);

– оператор отбора (вкладка Selection);

– оператор репродукции (вкладка Reproduction);

– оператор мутации (вкладка Mutation);

– оператор скрещивания (вкладка Crossover);

– перенесение особей между популяциями (вкладка Migration);

– специальные параметры алгоритма (вкладка Algorithm settings);

– задание гибридной функции (вкладка Hybrid function);

– задание критерия остановки алгоритма (вкладка Stopping criteria);

Таблица 4

Работа с Optimization Tool

Название и значение параметра в среде Matlab	**Программная реализация текущего параметра авторами**

В поле **Fitness function** указывается оптимизируемая функция	c = ones(1,21);
	betaA = diag(ones(7,1)) + diag(ones(6,1),-1) + diag(ones(5,1),-2) + diag(ones(4,1),-3) + diag(1,6) + diag([1;1],5) + diag(ones(3,1),4);
	A = -blkdiag(betaA, betaA, betaA);
	b = – [5;3;2;4;3;2;2;7;8;9;5;7;2;5;9;10;10;7;11;2;2];
	lb = zeros(1,21);
	ub = 11*ones(1,21);
	schedule_fitness1 = @(x) c*x' + 20*sum(A*x'> b).

Во вкладке **Population** производится настройка работы с начальной популяцией	Используется собственная функция создания популяции:
	function Population = schedule_create(GenomeLength, FitnessFcn, options)
	totalpopulation = sum(options.PopulationSize);
	range = options.PopInitRange;
	lower = range(1,:);
	span = range(2,:) – lower;
	Population = repmat(lower,totalpopulation,1) + round(repmat(span,totalpopulation,1).* rand(totalpopulation,GenomeLength));
	End.

Вкладка **Reproduction** уточняет, каким образом происходит создание новых особей	Задается элитная выборка популяции (выжившие) как 10% от общей популяции. Задается значение доли скрещивания: 50% участвуют в скрещивании, 50% – в мутации.

Во вкладке оператора **Mutation** выбирается тип оператора мутации	Используется собственная функция мутации:
	function mutationChildren = schedule_mutate(parents, options, GenomeLength, FitnessFcn, state, thisScore, thisPopulation)
	lb = repmat(options.LinearConstr.lb', length(parents), 1);
	ub = repmat(options.LinearConstr.ub', length(parents), 1);
	RandChange = round(2.5*randn(length(parents), GenomeLength) – 0.05);
	mutationChildren = min(max(lb, thisPopulation(parents,:) + RandChange), ub);
	End.

Вкладка **Crossover** позволяет выбрать тип оператора скрещивания	Используется собственная функция скрещивания:
	function xoverKids = schedule_xover(parents, options, nvars, FitnessFcn, unused, thisPopulation)
	p1 = thisPopulation(parents(1:2:end),:);
	p2 = thisPopulation(parents(2:2:end),:);
	decide = rand(size(p1))<0.6;
	xoverKids = decide.* min(p1,p2) + ~decide.* max(p1,p2);
	End.

Во вкладке **Stopping criteria** указываются ситуации, при которых алгоритм совершает остановку	Указывается, что максимальное количество допустимых поколений составляет 1 000. Количество поколений, в течение которых фитнесс-функция не меняется, составляет 500. Укажем Function tolerance (минимальные значения изменений оптимизируемой функции, при которой алгоритм продолжит работу) как нуль.

– вывод различной дополнительной информации по ходу работы генетического алгоритма (вкладка Plot Functions);

– вывод результатов работы алгоритма в виде новой функции (вкладка Output function);

– задание набора информации для вывода в командное окно (вкладка Display to command window);

– способ вычисления значений оптимизированной и ограничивающей функций (вкладка User function evaluation).

Подробное описание настроек всех вышеперечисленных вкладок для нашей конкретной задачи, а также пользовательские m-функции, написанные авторами статьи, приведены в табл. 4.

После задания всех необходимых параметров утилита Optimization Tool выглядит так, как показано на рис. 2, из которого видно, что программа успешно завершила работу за 519 итераций.

Графическая интерпретация самого процесса оптимизации показана на рис. 3, из которого наглядно видны все основные процессы, происходящие в ГА при мутации, скрещивании и отборе нового лучшего поколения.

Результатом работы программы является матрица *X*, приведенная на рис. 4.

Поясним полученные результаты. Из рис. 4 видно, что три человека должны заступать в ночную смену в понедельник, два человека – в ночную смену в пятницу, один человек – в ночную смену в среду и так далее. Суммарное количество работников равно 33 (оптимальное значение фитнесс-функции).

Для сравнительного анализа и более глубокого понимания различий в работе ГА встроенной функции Matlab и этой же функции с инегрированным авторским программным кодом в работе [13] авторы приводили расчеты Matlab без учета их уникальных программных продуктов. Суммарное количество работников было равно 51 (оптимальное значение фитнесс-функции), однако данное решение было получено за значительно меньшее количество итераций (200).

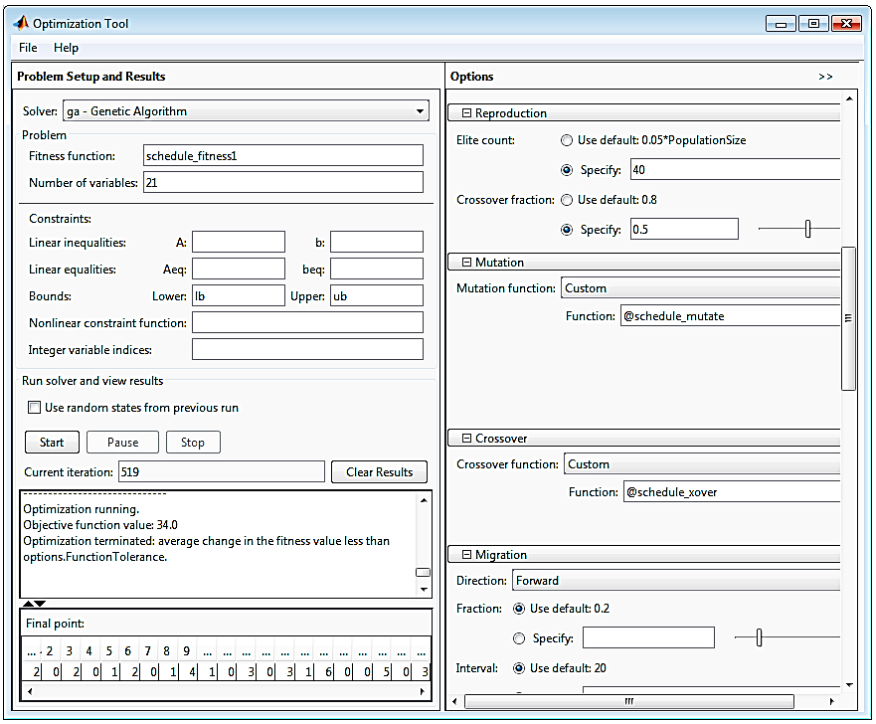


Рис. 2. Утилита Optimization Tool после того, как функция ga-GeneticAlgorithm отработала

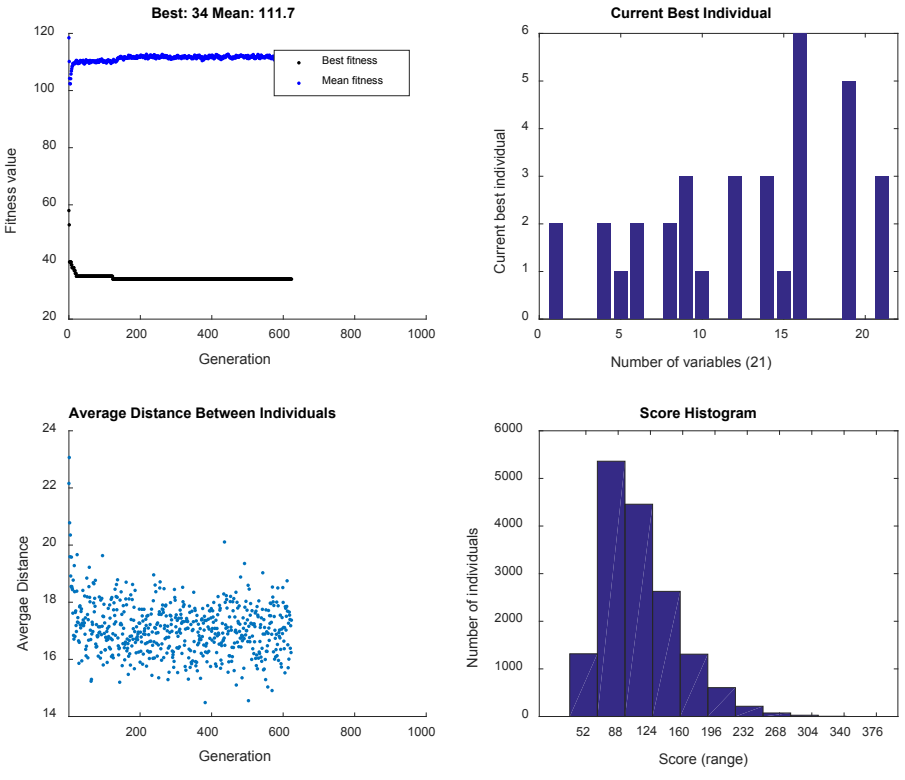
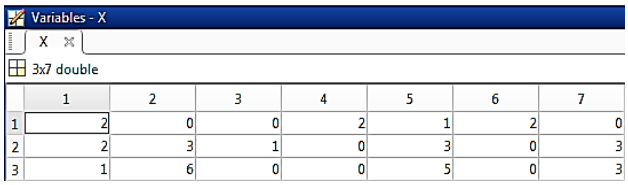


Рис. 3. Графическая интерпретация процесса оптимизации задачи о расписании с дополнительными ограничениями



	1	2	3	4	5	6	7
1	2	0	0	2	1	2	0
2	2	3	1	0	3	0	3
3	1	6	0	0	5	0	3

Рис. 4. Матрица значений фитнесс-функции

Выводы из проведенного исследования. В статье решена реальная экономическая задача теории расписаний одного из предприятий холдинга «Хлебные инвестиции». По типу искомого решения данная задача является задачей упорядочивания, по типу целевой функции – задачей с суммарным критерием оптимизации, по способу задания входной информации – задачей детерминированная. Решение представило собой временную таблицу, главной задачей ставилось не быстродействие, а минимизация количества работников предприятия (целочисленная задача линейного программирования). Было установлено, что данная задача по своей экономической сущности принадлежит к задачам без прерываний, что существенно затрудняет ее численное решение. Авторами статьи было установлено, что исходные данные этой задачи представляют собой матроид. Таким образом, согласно теореме Радо-Эдмондса, данная задача может быть решена с использованием генетических алгоритмов.

Проведенный анализ литературных источников показал, что ГА с успехом используются при решении задач составления расписаний, а также выявлены три основные этапа ГА, а именно скрещивание, селекция (отбор) и формирование нового поколения. Для всех основных этапов авторы статьи разработали свои уникальные m-функции и привели их в статье с открытым программным кодом, что придает работе колоссальную практическую ценность.

Для написания программного кода авторы статьи выбрали программную среду Matlab – мощный математический пакет, позволяющий максимально упростить процесс подготовки задачи, ее решения и анализа результатов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК:

1. Koffman E.G., Seti R., Bruno Dzh.L. Teoriya raspisaniy i vyichislitelnyie mashinyi / pod red. E.G. Koffmana. Moscow: Nauka, 1984. 334 s. URL: https://search.rsl.ru/ru/record/01001195670.

2. Wall B.W. A Genetic Algorithm for Resource Constrained Scheduling. PhD Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, USA. 1996. P. 62. URL: http://lancet.mit.edu/~mbwall/phd/thesis/thesis.pdf.https://www.coursehero.com/file/21647139/GA-Resource-Constrained-Scheduling-Thesis-Matthew-Wall.

3. Tryindyuk Yu.G. Tsar Hlib. URL: http://hlebinvest.com.ua/ru/about.

4. Rado R. A theorem on independence relations. Quart. J. Math. 1942. Vol. 13. P. 83-89. URL: https://doi.org/10.1093/qmath/os-13.1.83.

5. Yang S.-J., Hsu C.-J., Yang D.-L. Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time. Information Sciences: an International Journal. 2014. Vol. 260. P. 215-217. DOI: 10.1016/j.ins.2013.10.034.

6. Hsieh P.-H., Yang S.-J., Yang D.-L. Decision support for unrelated parallel machine scheduling with discrete controllable processing times. Applied Soft Computing. 2015. Vol. 30. P. 475-483. DOI: 10.1016/j.asoc. 2015.01.028.

7. Lazarev A.A., Musatova E.G., Tarasov I.A. Two-directional traffic scheduling problem solution for a single-track railway with siding. Automation and Remote Control. 2016. Vol. 77. № 12. P. 2118-2131. URL: https://doi.org/10.1134/S0005117916120031.

8. Sheibani K. An Incorporation of the Fuzzy Greedy Search Heuristic With Evolutionary Approaches for Combinatorial Optimization in Operations Management. International Journal of Applied Evolutionary Computation. Vol. 8. № 2. P. 58-72. DOI: 10.4018/IJAEC.2017040104.

9. He C., Leung J. Y.-T. Two-agent scheduling of time-dependent jobs. Journal of Combinatorial Optimization. 2017. Vol. 34. № 2. P. 362-377. DOI: 10.1007/s10878-016-9994-y.

10. Lim S.M., Sultan A.B. Md., Sulaiman Md. N., Mustapha A. Crossover and Mutation Operators of Genetic Algorithms. International Journal of Machine Learning and Computing. 2017. Vol. 7. № 1. P. 9-12. DOI: 10.18178/ijmlc. 2017.7.1.611.

11. Shamma M.N.E.-D.A., Shawki K.M., Bassioni H.A. Optimization of Construction Logistics Planning Cost in Egypt Using Genetic Algorithms. J Inform Tech Softw Eng. 2017. Vol. 7 (4): 205. P. 1-13. DOI: 10.4172/2165-7866.1000205.

12. Alharbi S., Venkat I. A Genetic Algorithm Based Approach for Solving the Minimum Dominating Set of Queens Problem. Journal of Optimization. 2017. Vol. 2017. P. 1-8. DOI: 10.1155/2017/5650364.

13. Kovaleva Ye.A., Misyura E.Yu. Zhadnoye postroyeniye raspisaniya v programmnoy srede MATLAB. Modern economic research: co-operation, banking, public administration in a decentralized environment: International Scientific Conference. Poland, Kielce, September 26th. 2017. P. 142-145. URL: http://www.repository.hneu.edu.ua/jspui/handle/123456789/18145.