Цель

Практическое закрепление теоретических основ информированного (эвристического) поиска с использованием продукционного программирования в среде CLIPS.

Постановка задачи

Рассматривается задача «Головоломка 8-ка».

Задана доска с 8 пронумерованными фишками и с одним пустым участком.

Фишка, смежная с пустым участком, может быть передвинута на этот участок. Требуется достичь указанного целевого состояния.

Начальное состояние:

3	**6**	**4**
2	**5**	**8**
7	**1**	

Целевое состояние:

	1	**2**
3	**4**	**5**
6	**7**	**8**

Разобрать по шагам (на некоторую глубину) реализацию алгоритма A* с использованием эвристических функций h_1 и h_2.

– h1 – число фишек, стоящих не на своем месте;

– h2 – суммарное по всем фишкам число шагов до целевого положения (манхэттенское расстояние).

Описание алгоритма

A*-поиск – разновидность **поиска сначала лучший (Best-First Seach – BFS)**, в котором порядок обхода вершин определяется эвристической функцией f (n) = g(n) + h(n), где h(n) – оценочная стоимость самого дешевого пути из состояния вершины n в целевое состояние (в случае Головоломки «Восьмерки» - это число фишек, стоящих не на своем месте или манхэттенское расстояние), g(h) – минимальная стоимость пути до узла n (в случае Головоломки «Восьмерки» - это глубина от начального состояния до состояния n).

Состояния представлены следующим образом:

LeftTop	MiddleTop	RightTop
LeftMiddle	MiddleMiddle	RightMiddle
LeftBottom	MiddleBottom	RightBottom

Рисунок 1

Описание выбранных структур данных

Для хранения информации о вершине была создана структура Node. В ней имеются идентификатор (Id), глубина (Depth), статус (Status), Id родителя (From), значение целевой функции (Exp), а также 9 слотов положений фишек, соответствующих рис.1 (LeftTop, MiddleTop, RightTop, LeftMiddle, MiddleMiddle, RightMiddle, LeftBottom, MiddleBottom, RightBottom). Идентификатор вершины назначается функцией Get_Id, увеличивающей его каждый раз при создании новой вершины.

Эвристическая функция W («Wrong position») возвращает количество цифр, находящихся не на своих местах.

Эвристическая функция M («Manhattan») возвращает сумму манхэттенских расстояний всех цифр от текущего положения до целевого.

Функция F считает целевую функцию в зависимости от выбранной эвристики (глобальная переменная H: 1-h1, 2-h2), складывая ее с глубиной вершины.

В качестве стартового факта создается факт о начальной вершине с заполненными полями и значение текущего минимума целевой функции.

Правила. Алгоритм

Очередность выполнения правил в среде CLIPS определяется приоритетом: чем он больше, тем раньше выполнится правило. Описание алгоритма:

1. Удаление повторов. Если найдена нераскрытая (статус 0) вершина u, имеющая такое же состояние, что и какая-то вершина v, а также значение целевой функции v меньше чем u, то удалить вершину (приоритет 1000).
2. Проверка целевого состояния. Если найдена вершина, чье состояние соответствует целевому, то она помечается как целевая (изменение статуса на 2) (приоритет 500).
3. Изменение статуса промежуточных вершин (бэктрекинг). Если найдена вершина со статусом 2, то статус ее родителя также изменяется на 2 (приоритет 500).
4. Удаление «не целевых» вершин. Если появилась вершина со статусом 2, то удаляем вершины со статусом, не равным 2 (приоритет 400).
5. Останов программы по нахождении решения. Если в базе фактов есть вершина с целевым состоянием, то остановить программу (приоритет 200).
6. Останов программы, если решение не существует. Если в базе фактов нет вершин, имеющих статус 0 или 2 (нераскрытых или целевых) – все вершины раскрыты, то остановить программу (приоритет 200).

7. Определение текущего минимума целевой функции. Если существует вершина, чья ЦФ меньше текущего минимума, то она становится минимумом (приоритет 150).
8. Понижение минимума, если нет нераскрытых вершин с таким значением ЦФ. Если не существует вершины, имеющей значение ЦФ, равное текущему минимуму, то минимум повышается на 1. Пример ситуации, в которой это может понадобится: вершина V1 имеет ЦФ=20, она раскрывается -> вершина V2 (ЦФ 21) и вершина V3 (ЦФ 22). После их добавления минимум обновится до значения 21, как минимального значения ЦФ из них. Однако, оказывается, что вершина V2 является повтором, она удаляется. Имеем: min 21, а вершин со статусом 0 и таким (или меньшим) ЦФ в базе фактов нет. Программа останавливается. Это правило позволяет избежать внезапного прекращения поиска (приоритет 120).
9. Правила перехода состояний (9). Для каждого из типов состояний (LeftTop=0, MiddleTop=0 и т.д.) определены состояния, в которые можно перейти, и рассчитывается ЦФ для каждой вершины (приоритет 100).

Исходный код

```
(deftemplate Node
(slot LeftTop(type NUMBER)) ;; 9 слотов для положения фишек
(slot MiddleTop(type NUMBER))
(slot RightTop(type NUMBER))
(slot LeftMiddle(type NUMBER))
(slot MiddleMiddle(type NUMBER))
(slot RightMiddle(type NUMBER))
(slot LeftBottom (type NUMBER))
(slot MiddleBottom(type NUMBER))
(slot RightBottom(type NUMBER))
(slot Depth(type NUMBER)) ;; глубина вершины в дереве
(slot Id(type NUMBER) (default 0)) ;; уникальный идентификатор вершины
(slot Status(type NUMBER) (default 0)) ;; статус вершины:

(slot From (type NUMBER)) ;; ссылка на родителя
```

```
(slot Exp (type NUMBER)) ;; значение целевой функции для данной вершины
);

(defglobal
?*Id* = 0 ;; объявляем и инициализируем глобальную переменную
?*H* = 1 ;;1 - h1, 2 - h2
?*STEP* = 1 ;;1 для пошагового режима
?*DEL* = 0 ;; количество удаленных повторных вершин
);

(deffunction Get_Id() ;; функция получения следующего Id
(bind ?*Id* (+ ?*Id* 1)) ;; инкрементируем Id
(return ?*Id*);
);

(deffunction W(
?LeftTop ?MiddleTop ?RightTop
?LeftMiddle ?MiddleMiddle ?RightMiddle
?LeftBottom ?MiddleBottom ?RightBottom)
(bind ?a 0)
(if (not (= ?LeftTop 0)) then (bind ?a (+ 1 ?a))) ;; подсчет f(n)
(if (not (= ?MiddleTop 1)) then (bind ?a (+ 1 ?a))) ;; в части h(n)
(if (not (= ?RightTop 2)) then (bind ?a (+ 1 ?a))) ;; по всем (8-ми!)
(if (not (= ?LeftMiddle 3)) then (bind ?a (+ 1 ?a)))
(if (not (= ?MiddleMiddle 4)) then (bind ?a (+ 1 ?a)))
(if (not (= ?RightMiddle 5)) then (bind ?a (+ 1 ?a))) ;; ячейкам
(if (not (= ?LeftBottom 6)) then (bind ?a (+ 1 ?a)))
(if (not (= ?MiddleBottom 7)) then (bind ?a (+ 1 ?a)))
(if (not (= ?RightBottom 8)) then (bind ?a (+ 1 ?a)))

(return ?a); ;; возвращаемое значение
);

(deffunction M(
?LeftTop ?MiddleTop ?RightTop
?LeftMiddle ?MiddleMiddle ?RightMiddle
?LeftBottom ?MiddleBottom ?RightBottom)
(bind ?a 0)

;;MiddleTop
(if (or (= ?LeftTop 1)(= ?MiddleMiddle 1)(= ?RightTop 1)) then (bind ?a (+ 1 ?a))
else (if (or(= ?LeftMiddle 1)(= ?MiddleBottom 1)(= ?RightMiddle 1)) then (bind ?a (+ 2 ?a)))
```

```
        else (if (or(= ?RightBottom 1)(= ?LeftBottom 1)) then (bind ?a (+ 3 ?a))))
        ;;RightTop
        (if (or (= ?MiddleTop 2)(= ?RightMiddle 2)) then (bind ?a (+ 1 ?a))
        else (if (or(= ?MiddleMiddle 2)(= ?RightBottom 2)(= ?LeftTop 2)) then (bind
?a (+ 2 ?a)))
        else (if (or(= ?LeftMiddle 2)(= ?MiddleBottom 2)) then (bind ?a (+ 3 ?a)))
        else (if (= ?LeftBottom 2) then (bind ?a (+ 4 ?a))))
        ;;LeftMiddle
        (if (or (= ?LeftTop 3)(= ?MiddleMiddle 3)(= ?LeftBottom 3)) then (bind ?a
(+ 1 ?a))
        else (if (or(= ?MiddleTop 3)(= ?RightMiddle 3)(= ?MiddleBottom 3)) then
(bind ?a (+ 2 ?a)))
        else (if (or(= ?RightTop 3)(= ?RightBottom 3)) then (bind ?a (+ 3 ?a))))
        ;;MiddleMiddle
        (if (or (= ?LeftMiddle 4)(= ?MiddleTop 4)(= ?RightMiddle 4)(=
?MiddleBottom 4)) then (bind ?a (+ 1 ?a))
        else (if (not(= ?MiddleMiddle 4)) then (bind ?a (+ 2 ?a))))
        ;;RightMiddle
        (if (or (= ?RightTop 5)(= ?MiddleMiddle 5)(= ?RightBottom 5)) then (bind
?a (+ 1 ?a))
        else (if (or(= ?MiddleTop 5)(= ?LeftMiddle 5)(= ?MiddleBottom 5)) then
(bind ?a (+ 2 ?a)))
        else (if (or(= ?LeftTop 5)(= ?LeftBottom 5)) then (bind ?a (+ 3 ?a))))
        ;;LeftBottom
        (if (or (= ?MiddleBottom 6)(= ?LeftMiddle 6)) then (bind ?a (+ 1 ?a))
        else (if (or(= ?MiddleMiddle 6)(= ?LeftTop 6)(= ?RightBottom 6)) then (bind
?a (+ 2 ?a)))
        else (if (or(= ?RightMiddle 6)(= ?MiddleTop 6)) then (bind ?a (+ 3 ?a)))
        else (if (= ?RightTop 6) then (bind ?a (+ 4 ?a))))
        ;;MiddleBottom
        (if (or (= ?LeftBottom 7)(= ?MiddleMiddle 7)(= ?RightBottom 7)) then (bind
?a (+ 1 ?a))
        else (if (or(= ?LeftMiddle 7)(= ?MiddleTop 7)(= ?RightMiddle 7)) then (bind
?a (+ 2 ?a)))
        else (if (or(= ?RightTop 7)(= ?LeftTop 7)) then (bind ?a (+ 3 ?a))))
        ;;RightBottom
        (if (or (= ?MiddleBottom 8)(= ?RightMiddle 8)) then (bind ?a (+ 1 ?a))
        else (if (or(= ?MiddleMiddle 8)(= ?RightTop 8)(= ?LeftBottom 8)) then (bind
?a (+ 2 ?a)))
        else (if (or(= ?LeftMiddle 8)(= ?MiddleTop 8)) then (bind ?a (+ 3 ?a)))
        else (if (= ?LeftTop 8) then (bind ?a (+ 4 ?a))))
        (return ?a);
        );

        (deffunction F(?Depth
```

```
        ?LeftTop ?MiddleTop ?RightTop
        ?LeftMiddle ?MiddleMiddle ?RightMiddle
        ?LeftBottom ?MiddleBottom ?RightBottom)
        (bind ?a ?Depth)
        (if (= ?*H* 1) then
          (bind ?a (+ ?a (W ?LeftTop ?MiddleTop ?RightTop ?LeftMiddle
?MiddleMiddle ?RightMiddle ?LeftBottom ?MiddleBottom ?RightBottom)))
        )
        (if (= ?*H* 2) then
          (bind ?a (+ ?a (M ?LeftTop ?MiddleTop ?RightTop ?LeftMiddle
?MiddleMiddle ?RightMiddle ?LeftBottom ?MiddleBottom ?RightBottom)))
        )
        (return ?a);
        );

        (deffunction print(?Depth ?Exp
        ?LeftTop ?MiddleTop ?RightTop
        ?LeftMiddle ?MiddleMiddle ?RightMiddle
        ?LeftBottom ?MiddleBottom ?RightBottom)

        (printout t "F=" ?Exp ", H=" (- ?Exp ?Depth) crlf
           ?LeftTop " " ?MiddleTop " " ?RightTop crlf
                ?LeftMiddle " " ?MiddleMiddle " " ?RightMiddle crlf
                ?LeftBottom " " ?MiddleBottom " " ?RightBottom crlf
         )
        )

        (deffacts start ;; факты соответствующие исходному состоянию
        (Node(LeftTop 3) (MiddleTop 6) (RightTop 4)
        (LeftMiddle 2) (MiddleMiddle 5)(RightMiddle 8)
        (LeftBottom 0) (MiddleBottom 1) (RightBottom 7)
        (Depth 0)
        (From 0)
        (Exp (F 0 3 6 4 2 5 8 0 1 7))
        (Id (Get_Id))
        )
        (min (F 0 3 6 4 2 5 8 0 1 7)) ;; фиксируется текущее значение min f(n)
        (if (= ?*STEP* 1) then (printout t crlf "Start node: " crlf)(print 0 (F 0 3 6 4 2
5 8 0 1 7) 3 6 4 2 5 8 0 1 7))
        );

        (defrule move_circle
        (declare (salience 1000)) ;; максимальный приоритет!!!
        ?f1<-(Node (LeftTop ?LT)(MiddleTop ?MT)(RightTop ?RT)
                        (LeftMiddle ?LM)(MiddleMiddle ?MM)(RightMiddle ?RM)
```

```
                    (LeftBottom ?LB)(MiddleBottom ?MB)(RightBottom
?RB)(Exp ?X)(Depth ?D1)) ;; первый факт
        ?f2<-(Node (LeftTop ?LT)(MiddleTop ?MT)(RightTop ?RT)
                    (LeftMiddle ?LM)(MiddleMiddle ?MM)(RightMiddle ?RM)
                    (LeftBottom ?LB)(MiddleBottom ?MB)(RightBottom
?RB)(Exp ?Y&~?X)(Status 0)(Depth ?D2));; второй факт
        ;;с неравным значением ЦФ
        (test(< ?X ?Y)) ;;большим, чем у первого состояния
        =>
        (retract ?f2) ;; удаление вершины с большей ЦФ
        (bind ?*DEL* (+ 1 ?*DEL*))
        (if (= ?*STEP* 1) then (printout t crlf "Delete repeating node: " crlf)(print
?D2 ?Y ?LT ?MT ?RT ?LM ?MM ?RM ?LB ?MB ?RB))
        )

        (defrule goal_test ;;проверка целевого состояния
        (declare (salience 500))
        ?f<-(Node (LeftTop 0) (MiddleTop 1) (RightTop 2);; состояние
        (LeftMiddle 3) (MiddleMiddle 4) (RightMiddle 5) ;; целевое
        (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
        (Status ~2) ;; текущий статус вершины «не целевое»
        (From ?Id)(Exp ?E))
        =>
        (modify ?f(Status 2));;текущая вершина помечается как «целевая»
        (printout t crlf "Solution: " crlf)
        )

        (defrule select_answer ;; изменение статуса промежуточных вершин
        ;; - бэктрекинг
        (declare (salience 500))
        (;; вершина со статусом 2, ссылается на
        Node(Status 2) (From ?Id)
        (LeftTop ?LT)(MiddleTop ?MT)(RightTop ?RT)
        (LeftMiddle ?LM)(MiddleMiddle ?MM)(RightMiddle ?RM)
        (LeftBottom ?LB)(MiddleBottom ?MB)(RightBottom ?RB)(Exp ?E)(Depth
?D)) ;; родителя
        ?f<-(Node(Id ?Id) (Status ~2))
        =>
        (modify ?f(Status 2))
        (print ?D ?E ?LT ?MT ?RT ?LM ?MM ?RM ?LB ?MB ?RB)
        )

        (defrule delete_not_answer
        (declare (salience 400)) ;; более низкий приоритет !!!
        (Node(Status 2)) ;; если появилась вершина со статусом 2
```

```
?f<-(Node(Status ~2)) ;; все вершина со статусом НЕ 2, надо удалить
=>
(retract ?f)
)

(defrule Stop_1 ;; выполняется, если решение не существует
(declare (salience 200)) ;; приоритет низкий
(not (Node(Status 0|2))) ;; в базе фактов нет вершин со статусом 0 или 2,
=> ;; т.е. нераскрытых и с целевым состоянием
(halt)
(printout t "No solutions" crlf)
)

(defrule Stop_2 ;; выполняется, если решение найдено
(declare (salience 200))
(Node(Status 2)) ;; есть вершина с целевым состоянием
=>
(halt)
(printout t "Found solution" crlf)
(printout t "Nodes count: " (- ?*Id* ?*DEL*) crlf)
(printout t "Step count: " ?*Id* crlf)
)

(defrule find_min ;; определение текущего минимума ЦФ
(declare (salience 150)) ;; приоритет низкий!
?fmin<-(min ?min) ;; адрес факта, хранящего значение текущего
;; min целевой функции
(Node(Exp ?E&:(< ?E ?min))(Status 0)) ;;существование
;; вершины, у которой значение целевой функции
;; меньше текущего min
=>
(retract ?fmin) ;; удалить факт с текущим min
(assert (min ?E)) ;; создать факт с новым текущим min
(if (= ?*STEP* 1) then (printout t crlf "New min: " ?min crlf))
)

(defrule min_not_found ;; повышение минимума, если нет вершин с такой
ЦФ
(declare (salience 120)) ;; приоритет низкий!
?fmin<-(min ?min) ;; адрес факта, хранящего значение текущего
;; min целевой функции
(not(exists(Node(Exp ?E&:(= ?E ?min))(Status 0)))) ;;существование
;; вершины, у которой значение целевой функции
;; равно текущему min
=>
```

```
(retract ?fmin) ;; удалить факт с текущим min
(assert (min (+ ?min 1))) ;; создать факт с новым min
(if (= ?*STEP* 1) then (printout t crlf "New min: " (+ ?min 1) crlf))
)

(defrule make_new_path_LeftTop
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
(LeftTop 0) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E 0 ?MT ?RT ?LM ?MM ?RM ?LB ?MB ?RB))
(bind ?a (F (+ 1 ?L) ?MT 0 ?RT ?LM ?MM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?MT) (MiddleTop 0) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a ?MT 0 ?RT ?LM ?MM ?RM ?LB ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LM ?MT ?RT 0 ?MM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LM) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle 0) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L)  ?a1 ?LM ?MT ?RT 0 ?MM ?RM ?LB ?MB ?RB))

(retract ?fmin)
(assert (min (min ?a ?a1)))
)

(defrule make_new_path_MiddleTop
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
```

```
(LeftTop ?LT) (MiddleTop 0) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT
0 ?RT ?LM ?MM ?RM ?LB ?MB ?RB))

(bind ?a (F (+ 1 ?L) 0 ?LT ?RT ?LM ?MM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop 0) (MiddleTop ?LT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a 0
?LT ?RT ?LM ?MM ?RM ?LB ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?RT 0 ?LM ?MM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?RT) (RightTop 0)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1
?LT ?RT 0 ?LM ?MM ?RM ?LB ?MB ?RB))

(bind ?a2 (F (+ 1 ?L) ?LT ?MM ?RT ?LM 0 ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MM) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle 0) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a2) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a2
?LT ?MM ?RT ?LM 0 ?RM ?LB ?MB ?RB))

(retract ?fmin)
(assert (min (min ?a ?a1 ?a2)))
)

(defrule make_new_path_RightTop
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
```

```
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop 0)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT ?MT 0 ?LM ?MM ?RM ?LB ?MB ?RB))

(bind ?a (F (+ 1 ?L) ?LT 0 ?MT ?LM ?MM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop 0) (RightTop ?MT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a ?LT 0 ?MT ?LM ?MM ?RM ?LB ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RM ?LM ?MM 0 ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RM)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle 0)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1 ?LT ?MT ?RM ?LM ?MM 0 ?LB ?MB ?RB))

(retract ?fmin)
(assert (min (min ?a ?a1)))
)

(defrule make_new_path_LeftMiddle
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle 0) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
```

```
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT
?MT ?RT 0 ?MM ?RM ?LB ?MB ?RB))

(bind ?a (F (+ 1 ?L) 0 ?MT ?RT ?LT ?MM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop 0) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LT) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a 0
?MT ?RT ?LT ?MM ?RM ?LB ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RT ?MM 0 ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?MM) (MiddleMiddle 0) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1
?LT ?MT ?RT ?MM 0 ?RM ?LB ?MB ?RB))

(bind ?a2 (F (+ 1 ?L) ?LT ?MT ?RT ?LB ?MM ?RM 0 ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LB) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom 0) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a2) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a2
?LT ?MT ?RT ?LB ?MM ?RM 0 ?MB ?RB))

(retract ?fmin)
(assert (min (min ?a ?a1 ?a2)))
)

(defrule make_new_path_MiddleMiddle
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым
полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle 0) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
```

```
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT
?MT ?RT ?LM 0 ?RM ?LB ?MB ?RB))

(bind ?a (F (+ 1 ?L) ?LT 0 ?RT ?LM ?MT ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop 0) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MT) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a
?LT 0 ?RT ?LM ?MT ?RM ?LB ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RT 0 ?LM ?RM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle 0) (MiddleMiddle ?LM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1
?LT ?MT ?RT 0 ?LM ?RM ?LB ?MB ?RB))

(bind ?a2 (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?RM 0 ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?RM) (RightMiddle 0)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a2) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a2
?LT ?MT ?RT ?LM ?RM 0 ?LB ?MB ?RB))

(bind ?a3 (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MB ?RM ?LB 0 ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MB) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom 0)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a3) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a3
?LT ?MT ?RT ?LM ?MB ?RM ?LB 0 ?RB))

(retract ?fmin)
(assert (min (min ?a ?a1 ?a2 ?a3)))
)
```

```
(defrule make_new_path_RightMiddle
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle 0)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT ?MT ?RT ?LM ?MM 0 ?LB ?MB ?RB))

(bind ?a (F (+ 1 ?L) ?LT ?MT 0 ?LM ?MM ?RT ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop 0)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RT)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a ?LT ?MT 0 ?LM ?MM ?RT ?LB ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RT ?LM 0 ?MM ?LB ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle 0) (RightMiddle ?MM)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1 ?LT ?MT ?RT ?LM 0 ?MM ?LB ?MB ?RB))

(bind ?a2 (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MM ?RB ?LB ?MB 0))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RB)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom 0)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a2) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a2 ?LT ?MT ?RT ?LM ?MM ?RB ?LB ?MB 0))

(retract ?fmin)
(assert (min (min ?a ?a1 ?a2)))
)
```

```
(defrule make_new_path_LeftBottom
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom 0) (MiddleBottom ?MB) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT ?MT ?RT ?LM ?MM ?RM 0 ?MB ?RB))

(bind ?a (F (+ 1 ?L) ?LT ?MT ?RT 0 ?MM ?RM ?LM ?MB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle 0) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LM) (MiddleBottom ?MB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a ?LT ?MT ?RT 0 ?MM ?RM ?LM ?MB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MM ?RM ?MB 0 ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?MB) (MiddleBottom 0)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1 ?LT ?MT ?RT ?LM ?MM ?RM ?MB 0 ?RB))

(retract ?fmin)
(assert (min (min ?a ?a1)))
)

(defrule make_new_path_MiddleBottom
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
```

```
(LeftBottom ?LB) (MiddleBottom 0) (RightBottom ?RB)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT
?MT ?RT ?LM ?MM ?RM ?LB 0 ?RB))

(bind ?a (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MM ?RM 0 ?LB ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom 0) (MiddleBottom ?LB)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a
?LT ?MT ?RT ?LM ?MM ?RM 0 ?LB ?RB))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RT ?LM 0 ?RM ?LB ?MM ?RB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle 0) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MM)(RightBottom ?RB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1
?LT ?MT ?RT ?LM 0 ?RM ?LB ?MM ?RB))

(bind ?a2 (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MM ?RM ?LB ?RB 0))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?RB)(RightBottom 0)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a2) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a2
?LT ?MT ?RT ?LM ?MM ?RM ?LB ?RB 0))

(retract ?fmin)
(assert (min (min ?a ?a1 ?a2)))
)

(defrule make_new_path_RightBottom
(declare (salience 100)) ;; приоритет самый низкий!!
?fmin<-(min ?min) ;; получение ссылки на факт с текущим минимумом
?f<-(Node(Status 0) (Depth ?L) (Id ?Id) ;; определение состояния с пустым
полем
(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
```

```
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom ?MB) (RightBottom 0)
(Exp ?E& :(= ?E ?min))) ;; проверка ЦФ вершины на min
=>
;;(printout t ?min " " (fact-slot-value ?f Exp) crlf)
(modify ?f(Status 1)) ;; изменение статуса вершины на «раскрыта»
(if (= ?*STEP* 1) then (printout t crlf "Current node: " crlf)(print ?L ?E ?LT ?MT ?RT ?LM ?MM ?RM ?LB ?MB 0))

(bind ?a (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MM 0 ?LB ?MB ?RM))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle 0)
(LeftBottom ?LB) (MiddleBottom ?MB)(RightBottom ?RM)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a ?LT ?MT ?RT ?LM ?MM 0 ?LB ?MB ?RM))

(bind ?a1 (F (+ 1 ?L) ?LT ?MT ?RT ?LM ?MM ?RM ?LB 0 ?MB))

(assert (Node(LeftTop ?LT) (MiddleTop ?MT) (RightTop ?RT)
(LeftMiddle ?LM) (MiddleMiddle ?MM) (RightMiddle ?RM)
(LeftBottom ?LB) (MiddleBottom 0)(RightBottom ?MB)
(Depth(+ ?L 1)) (From ?Id) (Exp ?a1) (Id (Get_Id))))
(if (= ?*STEP* 1) then (printout t crlf "New node: " crlf)(print (+ 1 ?L) ?a1 ?LT ?MT ?RT ?LM ?MM ?RM ?LB 0 ?MB))

(retract ?fmin)
(assert (min (min ?a ?a1)))
)
```

Результат работы, включая экспериментальные оценки временной (количества шагов) и емкостной (единиц памяти) сложности для своего варианта.

Найдем решение для начального состояния по варианту. Решение не найдено, по аналогии с Л.Р. 1,2.

Поменяв местами цифры 0 и 7 в начальном состоянии, найдем решение. Эвристика h1, пошаговый вывод:

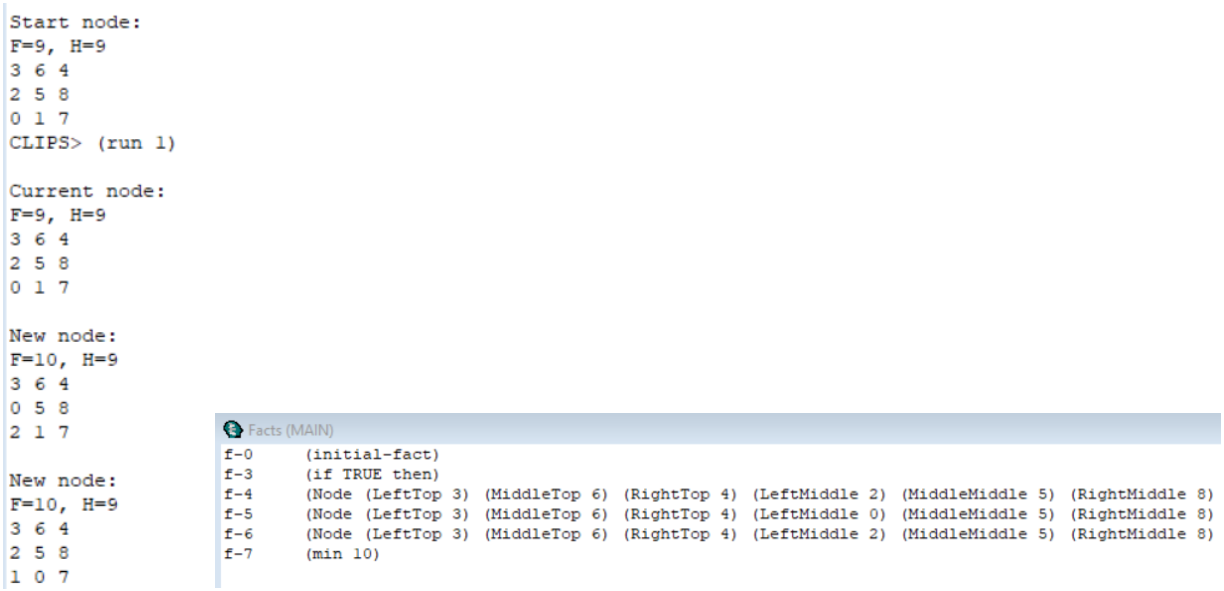

Рисунок 1. Шаг 1. Раскрытие стартовой вершины

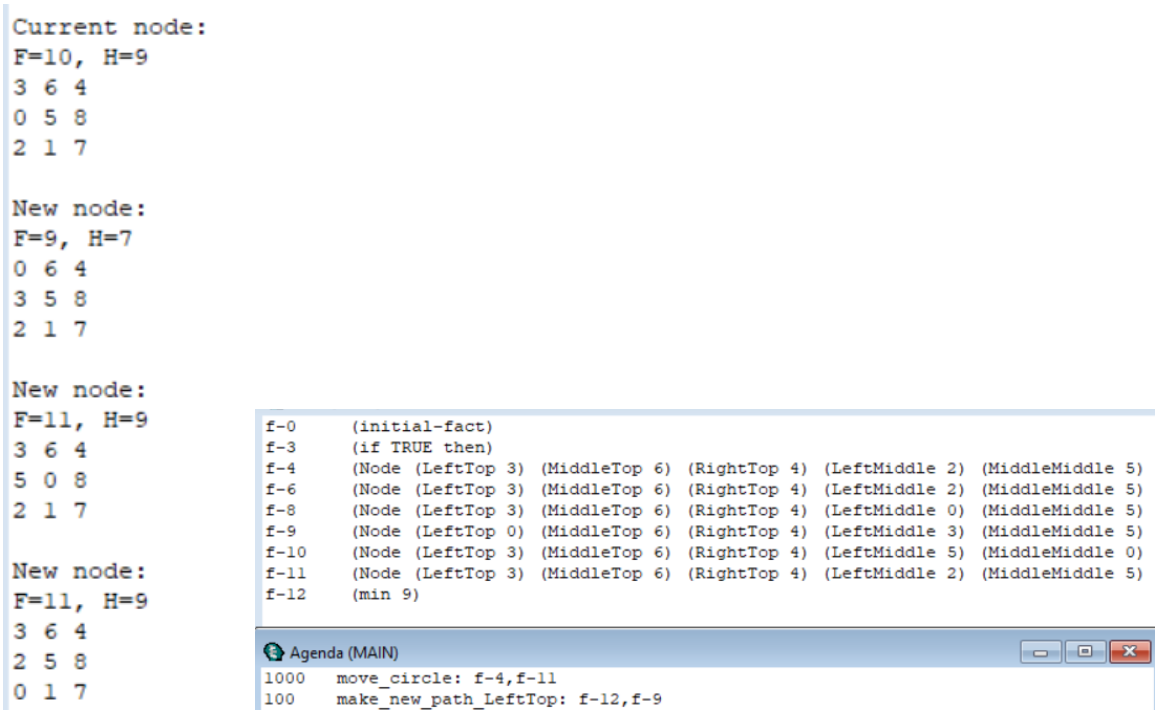

Рисунок 2. Шаг 2. Раскрытие вершины с минимальной ЦФ

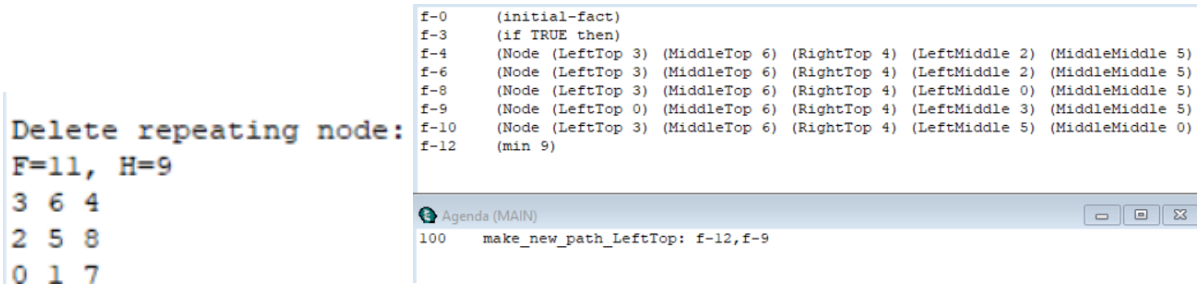

Рисунок 3. Шаг 3. Удаление повторной вершины

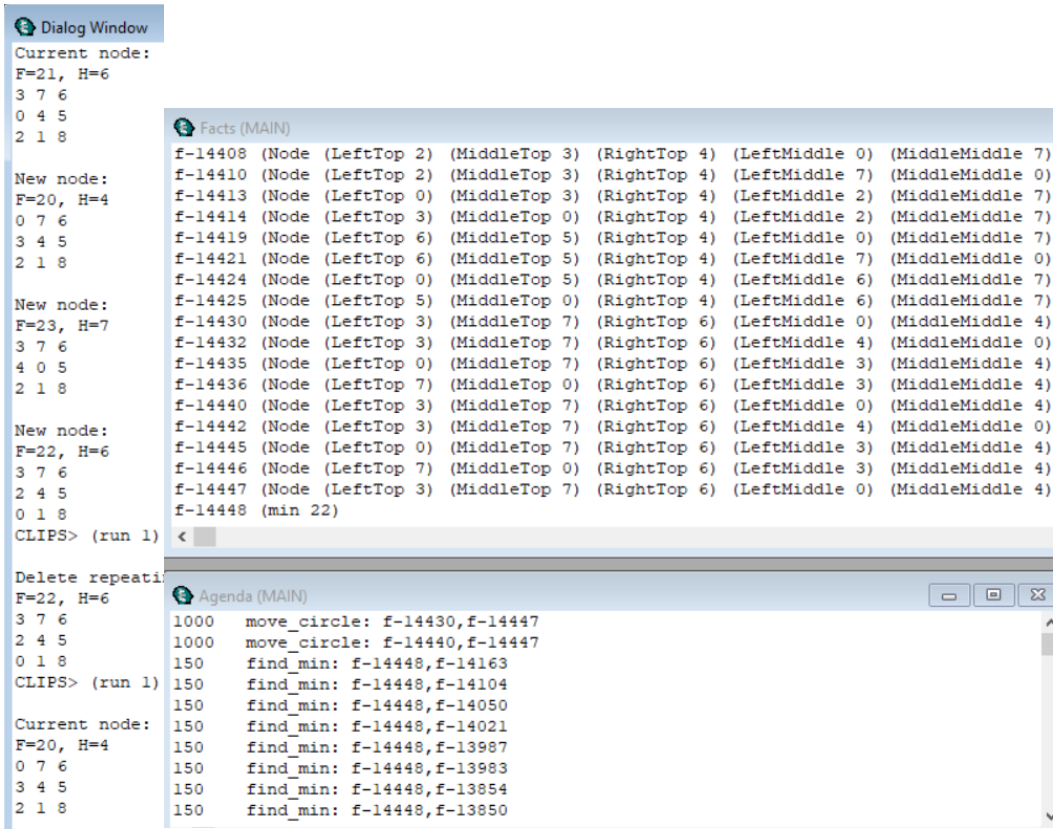

Рисунок 4. Процесс поиска. Выбор вершины для раскрытия

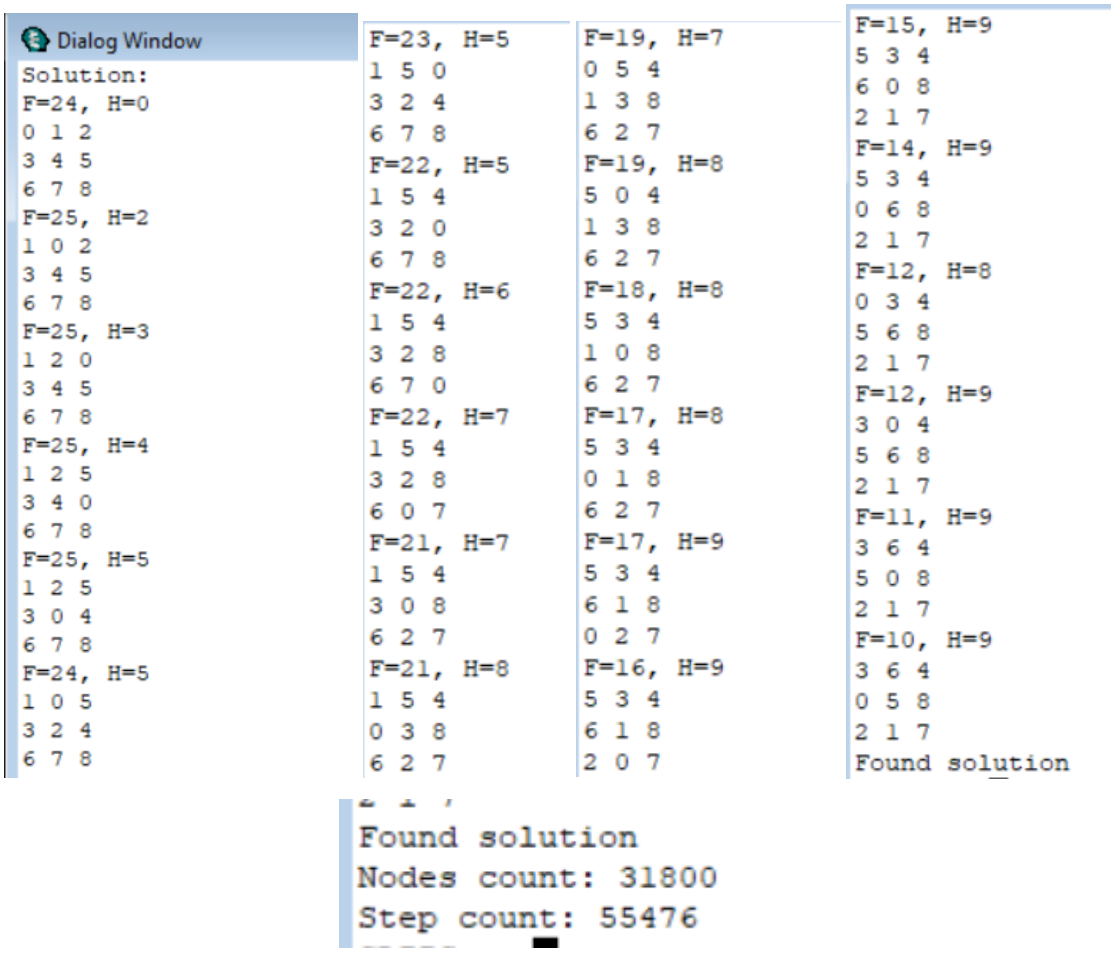

Рисунок 5. Вершины, составляющие путь от начального состояния до цели

```
f-0     (initial-fact)
f-3     (if TRUE then)
f-110753 (min 24)
f-110754 (Node (LeftTop 0) (MiddleTop 1) (RightTop 2) (LeftMiddle 3) (MiddleMiddle 4) (RightMiddle 5) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110755 (Id 55477)
f-110756 (Node (LeftTop 1) (MiddleTop 0) (RightTop 2) (LeftMiddle 3) (MiddleMiddle 4) (RightMiddle 5) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110757 (Node (LeftTop 1) (MiddleTop 2) (RightTop 0) (LeftMiddle 3) (MiddleMiddle 4) (RightMiddle 5) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110758 (Node (LeftTop 1) (MiddleTop 2) (RightTop 5) (LeftMiddle 3) (MiddleMiddle 4) (RightMiddle 0) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110759 (Node (LeftTop 1) (MiddleTop 2) (RightTop 5) (LeftMiddle 3) (MiddleMiddle 0) (RightMiddle 4) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110760 (Node (LeftTop 1) (MiddleTop 0) (RightTop 5) (LeftMiddle 3) (MiddleMiddle 2) (RightMiddle 4) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110761 (Node (LeftTop 1) (MiddleTop 5) (RightTop 0) (LeftMiddle 3) (MiddleMiddle 2) (RightMiddle 4) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110762 (Node (LeftTop 1) (MiddleTop 5) (RightTop 4) (LeftMiddle 3) (MiddleMiddle 2) (RightMiddle 0) (LeftBottom 6) (MiddleBottom 7) (RightBottom 8)
f-110763 (Node (LeftTop 1) (MiddleTop 5) (RightTop 4) (LeftMiddle 3) (MiddleMiddle 2) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 7) (RightBottom 0)
f-110764 (Node (LeftTop 1) (MiddleTop 5) (RightTop 4) (LeftMiddle 3) (MiddleMiddle 2) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 0) (RightBottom 7)
f-110765 (Node (LeftTop 1) (MiddleTop 5) (RightTop 4) (LeftMiddle 3) (MiddleMiddle 0) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 2) (RightBottom 7)
f-110766 (Node (LeftTop 1) (MiddleTop 5) (RightTop 4) (LeftMiddle 0) (MiddleMiddle 3) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 2) (RightBottom 7)
f-110767 (Node (LeftTop 0) (MiddleTop 5) (RightTop 4) (LeftMiddle 1) (MiddleMiddle 3) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 2) (RightBottom 7)
f-110768 (Node (LeftTop 5) (MiddleTop 0) (RightTop 4) (LeftMiddle 1) (MiddleMiddle 3) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 2) (RightBottom 7)
f-110769 (Node (LeftTop 5) (MiddleTop 3) (RightTop 4) (LeftMiddle 1) (MiddleMiddle 0) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 2) (RightBottom 7)
f-110770 (Node (LeftTop 5) (MiddleTop 3) (RightTop 4) (LeftMiddle 0) (MiddleMiddle 1) (RightMiddle 8) (LeftBottom 6) (MiddleBottom 2) (RightBottom 7)
f-110771 (Node (LeftTop 5) (MiddleTop 3) (RightTop 4) (LeftMiddle 6) (MiddleMiddle 1) (RightMiddle 8) (LeftBottom 0) (MiddleBottom 2) (RightBottom 7)
f-110772 (Node (LeftTop 5) (MiddleTop 3) (RightTop 4) (LeftMiddle 6) (MiddleMiddle 1) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 0) (RightBottom 7)
f-110773 (Node (LeftTop 5) (MiddleTop 3) (RightTop 4) (LeftMiddle 6) (MiddleMiddle 0) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 1) (RightBottom 7)
f-110774 (Node (LeftTop 5) (MiddleTop 3) (RightTop 4) (LeftMiddle 0) (MiddleMiddle 6) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 1) (RightBottom 7)
f-110775 (Node (LeftTop 0) (MiddleTop 3) (RightTop 4) (LeftMiddle 5) (MiddleMiddle 6) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 1) (RightBottom 7)
f-110776 (Node (LeftTop 3) (MiddleTop 0) (RightTop 4) (LeftMiddle 5) (MiddleMiddle 6) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 1) (RightBottom 7)
f-110777 (Node (LeftTop 3) (MiddleTop 6) (RightTop 4) (LeftMiddle 5) (MiddleMiddle 0) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 1) (RightBottom 7)
f-110778 (Node (LeftTop 3) (MiddleTop 6) (RightTop 4) (LeftMiddle 0) (MiddleMiddle 5) (RightMiddle 8) (LeftBottom 2) (MiddleBottom 1) (RightBottom 7)
f-110779 (Node (LeftTop 3) (MiddleTop 6) (RightTop 4) (LeftMiddle 2) (MiddleMiddle 5) (RightMiddle 8) (LeftBottom 0) (MiddleBottom 1) (RightBottom 7)
```

```
(Depth 24) (Id 55474) (Status 2) (From 55472) (Exp 24))

(Depth 23) (Id 55472) (Status 2) (From 55469) (Exp 25))
(Depth 22) (Id 55469) (Status 2) (From 54175) (Exp 25))
(Depth 21) (Id 54175) (Status 2) (From 28053) (Exp 25))
(Depth 20) (Id 28053) (Status 2) (From 17590) (Exp 25))
(Depth 19) (Id 17590) (Status 2) (From 15798) (Exp 24))
(Depth 18) (Id 15798) (Status 2) (From 15796) (Exp 23))
(Depth 17) (Id 15796) (Status 2) (From 15624) (Exp 22))
(Depth 16) (Id 15624) (Status 2) (From 7529) (Exp 22))
(Depth 15) (Id 7529) (Status 2) (From 7496) (Exp 22))
(Depth 14) (Id 7496) (Status 2) (From 2307) (Exp 21))
(Depth 13) (Id 2307) (Status 2) (From 2303) (Exp 21))
(Depth 12) (Id 2303) (Status 2) (From 1872) (Exp 19))
(Depth 11) (Id 1872) (Status 2) (From 1288) (Exp 19))
(Depth 10) (Id 1288) (Status 2) (From 1285) (Exp 18))
(Depth 9) (Id 1285) (Status 2) (From 810) (Exp 17))
(Depth 8) (Id 810) (Status 2) (From 398) (Exp 17))
(Depth 7) (Id 398) (Status 2) (From 222) (Exp 16))
(Depth 6) (Id 222) (Status 2) (From 48) (Exp 15))
(Depth 5) (Id 48) (Status 2) (From 44) (Exp 14))
(Depth 4) (Id 44) (Status 2) (From 40) (Exp 12))
(Depth 3) (Id 40) (Status 2) (From 5) (Exp 12))
(Depth 2) (Id 5) (Status 2) (From 2) (Exp 11))
(Depth 1) (Id 2) (Status 2) (From 1) (Exp 10))
(Depth 0) (Id 1) (Status 2) (From 0) (Exp 9))
```

Рисунок 6. Вершины, составляющие решение

Временная сложность алгоритма - количество шагов по добавлению вершин – 55476. Емкостная сложность – количество раскрытых вершин – 31800.

Эвристика h2, пошаговый режим

```
Start node:
F=14, H=14
3 6 4
2 5 8
0 1 7
CLIPS> (run 1)

Current node:
F=14, H=14
3 6 4
2 5 8
0 1 7

New node:
F=16, H=15
3 6 4
0 5 8
2 1 7

New node:
F=16, H=15
3 6 4
2 5 8
1 0 7
```

```
f-0     (initial-fact)
f-3     (if TRUE then)
f-4     (Node (LeftTop 3) (MiddleTop 6) (RightTop 4) (LeftMiddle 2) (MiddleMiddle 5) (RightMiddle 8)
f-5     (Node (LeftTop 3) (MiddleTop 6) (RightTop 4) (LeftMiddle 0) (MiddleMiddle 5) (RightMiddle 8)
f-6     (Node (LeftTop 3) (MiddleTop 6) (RightTop 4) (LeftMiddle 2) (MiddleMiddle 5) (RightMiddle 8)
f-7     (min 16)
```

Agenda (MAIN)

```
100     make_new_path_LeftMiddle: f-7,f-5
100     make_new_path_MiddleBottom: f-7,f-6
```

Рисунок 7.

```
Solution:
F=24, H=0
0 1 2
3 4 5
6 7 8
F=24, H=1
1 0 2
3 4 5
6 7 8
F=24, H=2
1 4 2
3 0 5
6 7 8
F=24, H=3
1 4 2
3 5 0
6 7 8
F=24, H=4
1 4 0
3 5 2
6 7 8
F=24, H=5
1 0 4
3 5 2
6 7 8
```

```
F=22, H=4
0 1 4
3 5 2
6 7 8
F=22, H=5
3 1 4
0 5 2
6 7 8
F=22, H=6
3 1 4
5 0 2
6 7 8
F=22, H=7
3 1 4
5 2 0
6 7 8
F=22, H=8
3 1 4
5 2 8
6 7 0
F=22, H=9
3 1 4
5 2 8
6 0 7
```

```
F=22, H=10
3 1 4
5 0 8
6 2 7
F=22, H=11
3 0 4
5 1 8
6 2 7
F=22, H=12
0 3 4
5 1 8
6 2 7
F=22, H=13
5 3 4
0 1 8
6 2 7
F=22, H=14
5 3 4
6 1 8
0 2 7
F=22, H=15
5 3 4
6 1 8
2 0 7
```

```
F=22, H=16
5 3 4
6 0 8
2 1 7
F=22, H=17
5 3 4
0 6 8
2 1 7
F=20, H=16
0 3 4
5 6 8
2 1 7
F=18, H=15
3 0 4
5 6 8
2 1 7
F=18, H=16
3 6 4
5 0 8
2 1 7
F=16, H=15
3 6 4
0 5 8
2 1 7
Found solution
```

```
Found solution
Nodes count: 2412
Step count: 3985
```

Рисунок 8. Решение

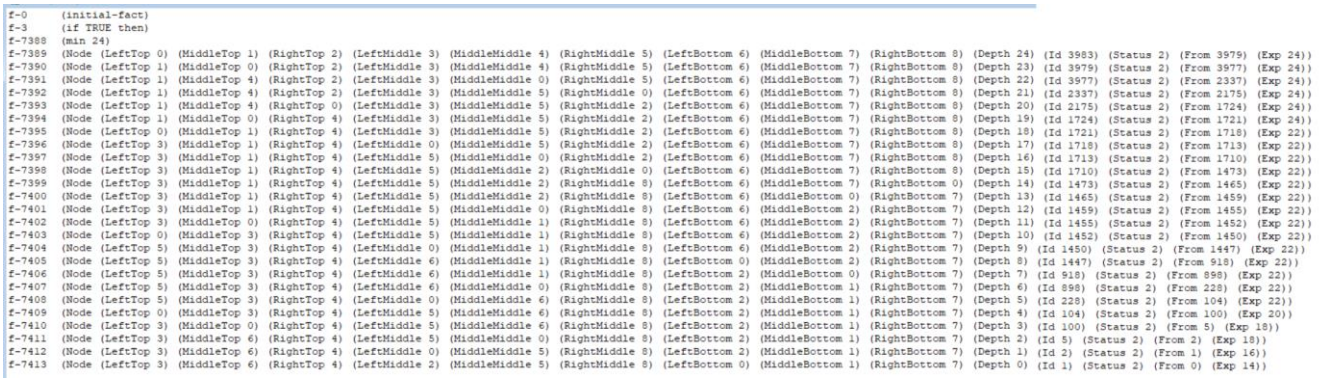

Рисунок 9. Вершины, составляющие решение

Временная сложность алгоритма - количество шагов по добавлению вершин – 3985. Емкостная сложность – количество раскрытых вершин – 2412.

Сравнение с результатами реализации эвристического поиска на языке императивного программирования

	Экспериментальная оценка временной сложности					
	Временная сложность (кол-во шагов)		Емкостная сложность (размер хэш-таблицы/кол-во вершин с уникальными состояниями в базе фактов)		Время выполнения	
	H1	H2	H1	H2	H1	H2
Java	49767	4877	27373	2890	4 с	0.1 с
CLIPS	55476	3985	31800	2412	≈70 мин	≈1 мин

Результаты реализации эвристического поиска на языке Java и в среде CLIPS близки по значению. Примерно одинаковы временная и емкостная сложность, потому что был применен одинаковый алгоритм, отличия вызваны разными подходы программирования (императивный и продукционный).

Время эвристического поиска сильно отличается, с тем учетом что тестировалось на одной вычислительной машине.

Это вызвано особенностями машины логических выводов среды CLIPS: вновь активируемые правила помещаются в агенду с учетом приоритета.

Для определения места среди правил равной значимости используется стратегия разрешения конфликта, по умолчанию используется алгоритм в глубину (вновь активируемые правила помещаются в агенду над всеми правилами такой же значимости).

По результатам обоих экспериментов видно, что программа при использовании эвристической функции h2 работает гораздо эффективнее и по временной, и по ёмкостной сложностям, чем при использовании h1. Это связано с тем, что эвристика h2 более информативна, чем h1. Состояния могут иметь одинаковое значение по h1, но разное по эвристике h2, более точно отражающей расстояние до целевого состояния.

Длины цепочек решений получились при использовании обеих эвристик – 24 узла.

Вывод

Закреплены теоретические основы информированного (эвристического) поиска с использованием продукционного программирования в среде CLIPS.