Шаховий рушій — оркестратор (повна розмова, код і інструкції)

Дата: 2025-08-13 (Європа/Київ)

Зміст

---

1) Вступ: еволюція алгоритмів у шахових ботах

Короткий конспект підходів від простих до складних (для контексту):

- **Жадібна оцінка**: $\mathrm{eval}(s) = \sum_i w_i\, \phi_i(s)$. Матеріал — як базова компонента.
- **Minimax / NegaMax**: повний перебір до глибини, складність $\mathcal{O}(b^d)$. Уніфікація через NegaMax.
- **Alpha–Beta**: відсікання гілок; з хорошим порядком — близько $\mathcal{O}(b^{d/2})$.
- **Quiescence**: дорозгляд тактичних ходів на листках.
- **Евристики**: MVV-LVA, killer/history, PST, iterative deepening, aspiration windows.
- **TT (Zobrist)**: кешування оцінок і меж.
- **PVS/LMR/Null-move**: сучасні прискорювачі в стилі Stockfish.
- **MCTS + NN (PUCT)**: Leela/AlphaZero підхід (політика + value мережі).
- **NNUE/комплексні eval**: швидка інкрементальна оцінка усередині αβ.

---

2) Чому в матеріальній оцінці немає короля

Король не має скінченної «ціни»: його «втрата» = кінець партії. Тому у **матеріалі** рахують лише обмінні ресурси (P,N,B,R,Q), а вплив короля йде через **безпеку** (штрафи за відкритість, атаки в зоні короля, відсутність «щитка» пішаків) та **активність** в ендшпілі (PST для King).

---

3) R: каркас NegaMax + αβ (базовий)

```
get_count <- function(counts, name) if (name %in% names(counts))
counts[[name]] else 0L

material_eval <- function(pos, weights = c(P = 100, N = 320, B = 330, R =
500, Q = 900)) {
  cts <- pos$counts
  weights["P"] * (get_count(cts, "P") - get_count(cts, "p")) +
    weights["N"] * (get_count(cts, "N") - get_count(cts, "n")) +
    weights["B"] * (get_count(cts, "B") - get_count(cts, "b")) +
    weights["R"] * (get_count(cts, "R") - get_count(cts, "r")) +
    weights["Q"] * (get_count(cts, "Q") - get_count(cts, "q"))
}

eval_position <- function(pos) material_eval(pos)

generate_moves_fn <- function(pos) list()
make_move_fn <- function(pos, mv) pos
is_terminal_fn <- function(pos) list(terminal = FALSE, score = 0)

negamax_ab <- function(pos, depth, alpha = -Inf, beta = Inf, color = 1L,
eval_fn = eval_position) {
  term <- is_terminal_fn(pos)
  if (term$terminal || depth == 0L) return(list(score = color *
eval_fn(pos), best_move = NULL))
  best_score <- -Inf; best_move <- NULL
  moves <- generate_moves_fn(pos)
  for (mv in moves) {
    child <- make_move_fn(pos, mv)
    res <- negamax_ab(child, depth - 1L, -beta, -alpha, -color, eval_fn)
    score <- -res$score
    if (score > best_score) { best_score <- score; best_move <- mv }
    if (best_score > alpha) alpha <- best_score
    if (alpha >= beta) break
  }
  list(score = best_score, best_move = best_move)
}
```

---

4) Python ⇄ R: приклад через `rpy2` (матеріал)

```
# pip install rpy2
import rpy2.robjects as ro
from rpy2.robjects.vectors import ListVector

ro.r("""
get_count <- function(counts, name) if (name %in% names(counts))
counts[[name]] else 0L
material_eval <- function(pos_counts) {
  w <- c(P=100,N=320,B=330,R=500,Q=900)
  cts <- pos_counts
  w["P"]*(get_count(cts,"P")-get_count(cts,"p")) +
  w["N"]*(get_count(cts,"N")-get_count(cts,"n")) +
  w["B"]*(get_count(cts,"B")-get_count(cts,"b")) +
  w["R"]*(get_count(cts,"R")-get_count(cts,"r")) +
  w["Q"]*(get_count(cts,"Q")-get_count(cts,"q"))
}
""")

counts = ListVector({'P':8,'N':2,'B':2,'R':2,'Q':1,'K':1,'p':8,'n':2,'b':
2,'r':2,'q':1,'k':1})
print('Material score =', ro.globalenv['material_eval'](counts)[0])
```

---

5) MCTS (PUCT): навчальний шаблон на Python

```
# pip install python-chess
import math, random, chess

class DummyNet:
    def predict(self, board):
        moves = list(board.legal_moves)
        if not moves: return {}, (-1.0 if board.is_checkmate() else 0.0)
        priors = [random.random() for _ in moves]; s = sum(priors)
        policy = {m.uci(): p/s for m,p in zip(moves, priors)}
        return policy, 0.0

class Node:
    __slots__ = ("board","parent","prior","children","N","W","Q","move_uci")
    def __init__(self, board, prior=0.0, parent=None, move_uci=None):
        self.board=board; self.parent=parent; self.prior=prior
        self.children={}; self.N=0; self.W=0.0; self.Q=0.0;
self.move_uci=move_uci
    def is_expanded(self): return bool(self.children)

class MCTS:
    def __init__(self, net, c_puct=1.5): self.net=net; self.c_puct=c_puct
```

```
    def search(self, root, n_simulations=200):
        for _ in range(n_simulations):
            node=root; path=[node]
            while node.is_expanded(): node=self._select(node);
path.append(node)
            if node.board.is_game_over(): v=self._terminal(node.board)
            else:
                pol,v=self.net.predict(node.board); self._expand(node, pol)
            self._backprop(path, v)
        return max(root.children.values(), key=lambda n:n.N).move_uci if
root.children else None
    def _select(self, node):
        S=sum(ch.N for ch in node.children.values()); best=-1e9; bc=None
        for ch in node.children.values():
            u=self.c_puct*ch.prior*math.sqrt(S+1e-8)/(1+ch.N); s=ch.Q+u
            if s>best: best=s; bc=ch
        return bc
    def _expand(self, node, policy):
        moves=list(node.board.legal_moves); ps=[policy.get(m.uci(),0.0) for m
in moves]
        s=sum(ps); pri=[p/s if s>0 else 1/len(moves) for p in ps]
        for m,p in zip(moves, pri):
            b=node.board.copy(); b.push(m);
node.children[m.uci()]=Node(b,p,node,m.uci())
    def _backprop(self, path, v):
        for n in reversed(path): n.N+=1; n.W+=v; n.Q=n.W/n.N; v=-v
    def _terminal(self, b): return -1.0 if b.is_checkmate() else 0.0
```

---

6) «Королівський» внесок у оцінку: формули та R-код

Фаза $\text{phase} \in [0,1]$: 1 — мітдлґейм, 0 — ендшпіль.

Оцінка:

$$ \text{eval} = \text{mat} + \text{pst_minor} + (\text{KA}_W-\text{KA}_B) - (\text{KS}_W-\text{KS}_B), $$

де $\text{KS} = c_{\text{check}}\,\mathbf{1}\{\text{check}\} + c_{\text{zone}}\sum_t w_t n_t(\text{king-zone}) + c_{\text{shelter}}\,\text{missing\_pawns}$, $\text{KA} = (1-\text{phase})\cdot \text{PST}_K^{\text{end}}[\text{king\_square}]$.

R-реалізація (скорочено, ядро):

```
phase_from_counts <- function(counts, max_nonking = 4000) { /* як раніше */ }
material_eval <- function(pos, w = c(P=100,N=320,B=330,R=500,Q=900)) { /* як
раніше */ }
king_safety_term <- function(side_feats, w_att=c(P=20,N=40,B=40,R=80,Q=160),
c_check=800, c_shelter=30, queen_scale=1.0) { /* ... */ }
king_activity_term <- function(king_sq, pst_endK) pst_endK[king_sq + 1L]
```

```
eval_position_complex <- function(pos, pst_endK = rep(0,64),
w_att=c(P=20,N=40,B=40,R=80,Q=160), c_check=800, c_shelter=30) {
  phase <- phase_from_counts(pos$counts)
  mat <- material_eval(pos)
  ks_w <- king_safety_term(pos$features$kings$white, w_att, c_check,
c_shelter, queen_scale = if (get_count(pos$counts,"q")>0) 1.2 else 1.0)
  ks_b <- king_safety_term(pos$features$kings$black, w_att, c_check,
c_shelter, queen_scale = if (get_count(pos$counts,"Q")>0) 1.2 else 1.0)
  ka_w <- (1 - phase) *
king_activity_term(pos$features$kings$white$king_square, pst_endK)
  ka_b <- (1 - phase) *
king_activity_term(pos$features$kings$black$king_square, pst_endK)
  unname(mat + (ka_w - ka_b) - (ks_w - ks_b))
}
```

---

7) Python ⇄ R (комплексна оцінка): приклад виклику

```
# pip install rpy2
import rpy2.robjects as ro
from rpy2.robjects.vectors import ListVector

ro.r("""
# (встав сюди eval_position_complex з розділу 6)
""")

counts = ListVector({'P':8,'N':2,'B':2,'R':2,'Q':1,'K':1,'p':8,'n':2,'b':
2,'r':2,'q':1,'k':1})
white_feats = ListVector({'in_check': False,'attacks_by_type':
ListVector({'P': 1,'N': 1,'B': 0,'R': 0,'Q': 0}), 'missing_shield': 1,
'king_square': 60})
black_feats = ListVector({'in_check': False,'attacks_by_type':
ListVector({'P': 0,'N': 0,'B': 1,'R': 0,'Q': 1}), 'missing_shield': 0,
'king_square': 4})
pos = ListVector({'counts': counts, 'features': ListVector({'kings':
ListVector({'white': white_feats, 'black': black_feats})})})
print('eval =', ro.globalenv['eval_position_complex'](pos)[0])
```

---

8) PyTorch policy-value з промоціями для `python-chess`

```
# pip install python-chess torch
import torch, torch.nn as nn, torch.nn.functional as F, chess

def board_to_planes(board: chess.Board):
    planes = torch.zeros(13,8,8)
    mp={chess.PAWN:0,chess.KNIGHT:1,chess.BISHOP:2,chess.ROOK:3,chess.QUEEN:
```

```
4,chess.KING:5}
    for sq,pc in board.piece_map().items():
        r,c=divmod(sq,8); ch=mp[pc.piece_type]+(0 if pc.color==chess.WHITE
else 6)
        planes[ch,7-r,c]=1.0
    planes[12,:,:]=1.0 if board.turn==chess.WHITE else 0.0
    return planes.unsqueeze(0)

def promo_index(m):
    return {None:0, chess.KNIGHT:1, chess.BISHOP:2, chess.ROOK:3,
chess.QUEEN:4}[m.promotion]

class PolicyValuePromo(nn.Module):
    def __init__(self, emb=128):
        super().__init__()
        self.conv=nn.Sequential(nn.Conv2d(13,64,3,padding=1),nn.ReLU(),
nn.Conv2d(64,64,3,padding=1),nn.ReLU())
        self.val_head=nn.Sequential(nn.Flatten(), nn.Linear(64*8*8, emb),
nn.ReLU(), nn.Linear(emb,1))
        self.board_emb=nn.Sequential(nn.Flatten(), nn.Linear(64*8*8, emb),
nn.ReLU())
        self.from_emb=nn.Embedding(64,32); self.to_emb=nn.Embedding(64,32);
self.promo_emb=nn.Embedding(5,16)
        self.move_fc=nn.Sequential(nn.Linear(emb+32+32+16, emb), nn.ReLU(),
nn.Linear(emb,1))
    def forward(self, planes, from_idx, to_idx, promo_idx):
        x=self.conv(planes); v=torch.tanh(self.val_head(x)).squeeze()
        h=self.board_emb(x); f=self.from_emb(from_idx);
t=self.to_emb(to_idx); p=self.promo_emb(promo_idx)
        logits=self.move_fc(torch.cat([h.expand(f.shape[0],-1), f,t,p],
dim=1)).squeeze(-1)
        return logits, v

class TorchNet:
    def __init__(self, model, device="cpu"):
self.model=model.to(device).eval(); self.device=device
    @torch.no_grad()
    def predict(self, board):
        moves=list(board.legal_moves)
        if not moves: return {}, (-1.0 if board.is_checkmate() else 0.0)
        planes=board_to_planes(board).to(self.device)
        fi=torch.tensor([m.from_square for m in
moves],dtype=torch.long,device=self.device)
        ti=torch.tensor([m.to_square for m in
moves],dtype=torch.long,device=self.device)
        pi=torch.tensor([promo_index(m) for m in
moves],dtype=torch.long,device=self.device)
        logits,val=self.model(planes,fi,ti,pi)
        p=torch.softmax(logits,dim=0).cpu().tolist()
        return {m.uci():pr for m,pr in zip(moves,p)}, float(val.item())
```

9) R: αβ + Transposition Table + Quiescence

(Скорочена версія — повний код у розділах 11–13.)

```
# TT/flags + qsearch + negamax_ab_tt(...) як у докладних розділах нижче
```

10) Python MCTS: Dirichlet noise + Temperature (self-play)

```
import numpy as np
class MCTS:
    # ... (базовий з розд. 5)
    def _add_dirichlet_noise(self, root, alpha=0.3, eps=0.25):
        K=len(root.children); noise=np.random.dirichlet([alpha]*K)
        for ch,n in zip(root.children.values(), noise): ch.prior=(1-
eps)*ch.prior+eps*float(n)
    def select_action(self, root, temperature=1.0):
        moves=list(root.children.keys()); counts=np.array([root.children[m].N
for m in moves], float)
        if temperature<=1e-6: return moves[int(np.argmax(counts))]
        probs=counts**(1.0/temperature); probs/=probs.sum()+1e-12
        return np.random.choice(moves, p=probs)
```

11) R: PVS + LMR (пізні редукції)

```
lmr_amount <- function(depth, move_idx, is_capture, gives_check, in_pv) {
  if (in_pv || depth < 3L || move_idx < 4L) return(0L)
  r <- floor(log(depth + 1) * log(move_idx + 1) / 2.0)
  if (isTRUE(gives_check)) r <- max(0L, r - 1L)
  as.integer(max(0L, min(r, depth - 1L)))
}
# negamax_pvs_lmr(...) — див. повний варіант у розд. 13
```

12) Python: батчова мережа (`predict_many`) + пакетний MCTS

```
import torch, numpy as np, chess

def boards_to_batch(boards):
    return torch.stack([board_to_planes(b) for b in boards], dim=0)

class PolicyValuePromo(nn.Module):
```

```
    # ... (як у розд. 8) + методи forward_values/forward_moves

class TorchNet:
    # ... (add predict_many)
    @torch.no_grad()
    def predict_many(self, boards):
        legal=[list(b.legal_moves) for b in boards]; B=len(boards)
        planes=boards_to_batch(boards).to(self.device)
        h,v=self.model.forward_values(planes)
        sizes=[len(mv) for mv in legal]
        if sum(sizes)==0:
            return [{} for _ in boards], [(-1.0 if b.is_checkmate() else 0.0)
for b in boards]
        bi=torch.tensor([i for i,sz in enumerate(sizes) for _ in
range(sz)],dtype=torch.long,device=self.device)
        fi=torch.tensor([m.from_square for mv in legal for m in
mv],dtype=torch.long,device=self.device)
        ti=torch.tensor([m.to_square for mv in legal for m in
mv],dtype=torch.long,device=self.device)
        pi=torch.tensor([promo_index(m) for mv in legal for m in
mv],dtype=torch.long,device=self.device)
        logits=self.model.forward_moves(h.index_select(0,bi), fi, ti,
pi).cpu()
        policies=[]; start=0; vals=v.cpu().tolist()
        for i,sz in enumerate(sizes):
            if sz==0: policies.append({}); continue
            sl=slice(start,start+sz);
probs=torch.softmax(logits[sl],dim=0).numpy().tolist()
            policies.append({m.uci():p for m,p in zip(legal[i], probs)});
start+=sz
        return policies, vals

class MCTS:
    # ... (додай search_batch з колекцією листків і predict_many)
```

---

13) R: Null-move pruning + Fail-soft + мікро-профайлер (повний вузол)

```
# Параметри, TT, orderer (MVV-LVA/killers/history), qsearch — як у попередніх
розділах
# Нижче — основний вузол із Null-move + Fail-soft + лічильниками у профайлері
negamax_pvs_lmr <- function(pos, depth, alpha = -Inf, beta = Inf, color =
1L, ply = 0L,
                           eval_fn = eval_position, use_tt = TRUE, use_qs =
TRUE,
                           history_env = new.env(hash = TRUE), killers =
list(k1=list(),k2=list()),
                           order_moves = NULL, in_pv = TRUE) {
```

```
  # ... повний код вузла з Null-move/LMR/PVS/Fail-soft/TT/профайлером (див.
чат)
}

search_iterative <- function(pos, max_depth = 10L, eval_fn = eval_position,
                             start_window = 25L, use_qs = TRUE, use_tt =
TRUE, verbose = TRUE) {
  # ... як у чаті (aspiration fail-soft + звіт профайлера)
}
```

---

14) R: one-shot пошук (без iterative deepening)

```
search_one_shot <- function(pos, depth, alpha = -Inf, beta = Inf,
                            eval_fn = eval_position, use_qs = TRUE, use_tt =
TRUE, verbose = TRUE) {
  prof_start(); on.exit({ prof_stop(); if (verbose) prof_report() }, add =
TRUE)
  history_env <- new.env(hash = TRUE)
  killers <- list(k1 = vector("list", 256L), k2 = vector("list", 256L))
  res <- negamax_pvs_lmr(pos, depth, alpha, beta, 1L, 0L, eval_fn, use_tt,
use_qs, history_env, killers, NULL, TRUE)
  if (verbose) cat(sprintf("depth=%d  score=%d  pv=%s\n", depth, res$score,
paste0(vapply(res$pv, mv_key, ""), collapse=" ")))
  res
}
```

---

15) Python: one-shot MCTS вибір ходу

```
def choose_move_one_shot(board, net, n_simulations=2048, batch_size=64,
                         c_puct=1.5, dirichlet_alpha=0.3, dirichlet_eps=0.25,
                         temperature=1.0, add_dirichlet=True):
    mcts = MCTS(net, c_puct=c_puct, dirichlet_alpha=dirichlet_alpha,
dirichlet_eps=dirichlet_eps)
    root = Node(board)
    return mcts.search_batch(root, n_simulations=n_simulations,
batch_size=batch_size,
                             add_dirichlet=add_dirichlet,
temperature=temperature)
```

---

16) Гібрид-оркестратор: MCTS → top-K → AB-валідація (R-eval)

R-оцінка (встав через ``): див. розд. 6.\ **Python-оркестратор (скорочено нижче; повна версія — у чаті):**

```
import chess, numpy as np
from rpy2.robjects.vectors import ListVector

# REvalBridge: будує counts/king-features для eval_position_complex та
викликає R
# ab_pvs: невеликий αβ+PVS+QSearch на python-chess з R-eval як статичка
# HybridOrchestrator.choose(...):
#  1) робить пакетний MCTS,
#  2) бере top-K за візитами,
#  3) для кожного кандидата рахує AB-оцінку на глибині verify_depth,
#  4) змішує MCTS та AB (λ-мікс) і повертає хід + діагностику
```

---

17) Візуалізація: графік порівняння MCTS/AB/Mix (matplotlib)

```
# pip install matplotlib
import numpy as np, matplotlib.pyplot as plt

def plot_orchestrator_diag(diag, title=None, save_path=None):
    cand=diag['candidates']
    moves=[c['move'] for c in cand]
    visits=np.array([c['visits'] for c in cand], float)
    mcts_norm=np.array([c['mcts_norm'] for c in cand], float)
    ab_score=np.array([c['ab_score'] for c in cand], float)
    ab_norm=np.array([c['ab_norm'] for c in cand], float)
    mix=np.array([c['mix'] for c in cand], float)
    idx=np.arange(len(moves)); chosen=diag.get('chosen')

    fig,axs=plt.subplots(1,3,figsize=(min(4+2.5*len(moves),18),
4),constrained_layout=True)
    colors=['#4e79a7' if m!=chosen else '#f28e2b' for m in moves]
    axs[0].bar(idx, visits, color=colors); axs[0].set_title('MCTS visits');
axs[0].set_ylabel('count'); axs[0].set_xticks(idx, moves, rotation=45,
ha='right')
    axs[1].bar(idx, ab_score, color=colors);
axs[1].axhline(0,color='k',lw=0.8); axs[1].set_title('AB score (cp)');
axs[1].set_ylabel('cp'); axs[1].set_xticks(idx, moves, rotation=45,
ha='right')
    w=0.27
    axs[2].bar(idx-w, mcts_norm, width=w, label='MCTS_norm', color='#59a14f')
    axs[2].bar(idx,   ab_norm,   width=w, label='AB_norm',   color='#e15759')
    axs[2].bar(idx+w, mix,       width=w, label='Mix',       color='#f28e2b')
    axs[2].set_ylim(0,1.05); axs[2].set_title('Normalized (0–1)');
axs[2].set_xticks(idx, moves, rotation=45, ha='right');
axs[2].legend(loc='upper right', frameon=False)
    if title: fig.suptitle(title, y=1.05)
    if save_path: plt.savefig(save_path, dpi=160, bbox_inches='tight')
    return fig, axs
```

18) Поради з тюнінгу та використання

- **R one-shot vs ID**: без iterative deepening порядок ходів гірший; компенсуйте TT, killers/history, aspiration-вікно з fail-soft.
- **Null-move (R)**: вмикайте в мітдлґеймі; обережно у «тонких» цугцванґових ендшпілях (verification search допоможе).
- **LMR (R)**: редукуйте пізні некепчери без шаху; перераховуйте повним вікном при scout-проривах.
- **MCTS batch**: під GPU обирайте `batch_size=64..256`; розносьте симуляції 1:1 з forward-часом.
- **Оркестратор**: обирайте `top_k=3..5`, `verify_depth=5..7`; керуйте `mix_lambda` динамічно (більше ваги AB у тактичних позиціях).

Нотатка про стиль R

Приклади коду дотримуються стилю: `snake_case`, пробіли навколо операторів, лаконічні коментарі «чому», явні імена, захист від `NULL` (`%||%`).

Кінець документа.