

УДК 004.42

ОПТИМИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ

Довнар С.С., Стальцова Е.А.

Научный руководитель – Прихожий А.А., д.т.н., профессор

Целью данной работы является разработка программного обеспечения для поиска оптимального решения задачи в игровом приложении, на примере игры «Волки и заяц».

Для решения поставленной задачи используется минимакс метод.

Введем некоторые понятия и обозначения:

- Состояние, оно же узел дерева решений — расположение фигур на доске.
- Терминальное состояние — состояние, с которого больше нет ходов (кто-то победил / ничья).
- V_i — i -ое состояние.
- V_{ik} — k -ое состояние, к которому можно прийти за один ход из состояния V_i . В контексте дерева решений, это дочерний узел V_i .
- $f(V_i)$ — расчетная оценка вероятности победы для состояния V_i .
- $g(V_i)$ — эвристическая оценка вероятности победы для состояния V_i .

для решения проблемы о выборе хода из состояния V_i разработан следующий метод. Метод выбирать ход, для которого оценка вероятности будет максимальной в плане выигрыша для игрока, который ходит. В этой метрике, при ходе со стороны волков — вероятность максимальная для волков и минимальная для зайцев. Оценка вероятности вычисляется следующим образом:

- $f(V_i) = g(V_i)$, если V_i — терминальное состояние, либо достигнут предел глубины расчетов.
- $f(V_i) = \max(f(V_{i1}), f(V_{i2}) \dots f(V_{ik}))$, если V_i — состояние, с которого ходит игрок с поиском максимальной оценки.
- $f(V_i) = \min(f(V_{i1}), f(V_{i2}) \dots f(V_{ik}))$, если V_i — состояние, с которого ходит игрок с поиском минимальной оценки.

Метод выбирает ход так, чтобы максимизировать вероятность P собственной победы, учитывая при этом как будет ходить противник, а противник будет ходить так, чтобы максимизировать вероятность своей победы и минимизировать вероятность P . Этот минимаксный подход выполняет оптимизацию со стороны одного игрока. Для этого вводятся два дополнительных параметра $alpha$ и $beta$, где $alpha$ — текущее максимальное значение, меньше которого игрок максимизации (волки) никогда не выберет, а $beta$ — текущее минимальное значение, больше которого игрок минимизации (заяц) никогда не выберет. В начале игры

они устанавливаются в $-\infty$ и $+\infty$ соответственно, и по мере получения оценок $f(V_i)$ модифицируются:

- $\alpha = \max(\alpha, f(V_i))$; для уровня максимизации.
- $\beta = \min(\beta, f(V_i))$; для уровня минимизации.

Как только условие $\alpha > \beta$ станет верным, наступает конфликт ожиданий, при этом анализ V_k прерывается и возвращается последняя полученная на этом уровне оценка.

Эффективность работы выше описанного метода была проверена на практике и получены результаты представленные на рисунках 1 и 2.



Рисунок 1 – Сравнение объема перебора алгоритма с оптимизацией и без нее

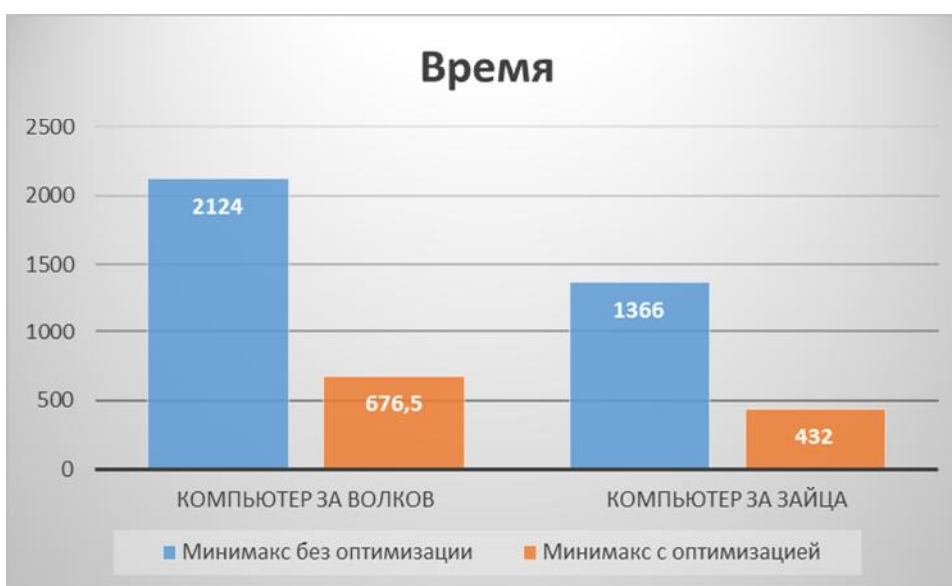


Рисунок 2– Сравнение времени работы с оптимизацией и без нее

Описанный метод реализован в виде программного обеспечения, написанного на языке программирования C++. К достоинствам настоящего проекта и реализованного алгоритма оптимизации относится:

- эффективность вычисления следующего хода – заведомо неудачные ходы отсекаются без прохода по соответствующим ветвям дерева поиска.
- охватываются все возможные варианты хода, как своего, так и соперника, а это обеспечивает выбор наилучшего хода.

Платой за высокое качество игрового алгоритма и разработанной программы является:

- возможное потребление значительного процессорного времени, что компенсируется применением альфа-бета отсечения;
- значительное потребление ресурсов памяти.