These simple approaches can lead to errors due to the approximate nature of the evaluation function. Consider again the simple evaluation function for chess based on material advantage. Suppose the program searches to the depth limit, reaching the position in Figure 5.8(13), where Black is ahead by a knight and two pawns. It would report this as the heuristic value of the state, thereby declaring that the state is a probable win by Black. But White’s next move captures Black’s queen with no compensation. Hence, the position is really won for White, but this can be seen only by looking ahead one more ply.

Obviously, a more sophisticated cutoff test is needed. The evaluation function should be applied only to positions that are quiescent—that is, unlikely to exhibit wild swings in value in the near future. In chess, for example, positions in which favorable captures can be made are not quiescent for an evaluation function that just counts material. Nonquiescent positions can be expanded further until quiescent positions are reached. This extra search is called a quiescence search; sometimes it is restricted to consider only certain types of moves, such as capture moves, that will quickly resolve the uncertainties in the position.

The horizon effect is more difficult to eliminate. It arises when the program is facing an opponent’s move that causes serious damage and is ultimately unavoidable, but can be temporarily avoided by delaying tactics. Consider the chess game in Figure 5.9. It is clear that there is no way for the black bishop to escape. For example, the white rook can capture it by moving to h1, then a1, then a2; a capture at depth 6 ply. But Black does have a sequence of moves that pushes the capture of the bishop "over the horizon." Suppose Black searches to depth 8 ply. Most moves by Black will lead to the eventual capture of the bishop, and thus will be marked as "bad" moves. But Black will consider checking the white king with the pawn at e4. This will lead to the king capturing the pawn. Now Black will consider checking again, with the pawn at f5, leading to another pawn capture. That takes up 4 ply, and from there the remaining 4 ply is not enough to capture the bishop. Black thinks that the line of play has saved the bishop at the price of two pawns, when actually all it has done is push the inevitable capture of the bishop beyond the horizon that Black can see.

One strategy to mitigate the horizon effect is the singular extension, a move that is "clearly heifer" than all other moves in a given position. Once discovered anywhere in the tree in the course of a search, this singular move is remembered. When the search reaches the normal depth limit, the algorithm checks to see if the singular extension is a legal move; if it is, the algorithm allows the move to be considered. This makes the tree deeper, but because there will be few singular extensions, it does not add many total nodes to the tree.

### 5.4.3 Forward pruning

So far, we have talked about cutting off search at a certain level and about doing alphabets pruning that provably has no effect on the result (at least with respect to the heuristic evaluation values). It is also possible to do forward pruning, meaning that some moves at a given node are pruned immediately without further consideration. Clearly, most humans playing chess consider only a few moves from each position (at least consciously). One approach to forward pruning is beam search: on each ply, consider only a "beam" of the a best moves (according to the evaluation function) rather than considering all possible moves.
Figure S. The horizon effect. With Black to move, the black bishop is surely doomed, but Black can forestall that event by checking the white king with its pawns, forcing the king to capture the pawns. This pushes the inevitable loss of the bishop over the horizon, and thus the pawn sacrifices are seen by the search algorithm as good moves rather than bad ones.

Unfortunately, this approach is rather dangerous because there is no guarantee that the best move will not be pruned away.

The **PROBCUT**, or probabilistic cut, algorithm (Buro, 1995) is a forward-pruning version of alpha—beta search that uses statistics gained from prior experience to lessen the chance that the best move will be pruned. Alpha—beta search prunes any node that is *provably* outside the current \((a, \beta/3)\) window. PROBCUT also prunes nodes that are *probably* outside the window. It computes this probability by doing a shallow search to compute the backed-up value \(v\) of a node and then using past experience to estimate how likely it is that a score of \(e\) at depth \(d\) in the tree would be outside \((a, \beta)\). Buro applied this technique to his Othello program, **LOGISTE110**, and found that a version of his program with **PROBCUT** beat the regular version 64% of the time, even when the regular version was given twice as much time.

Combining all the techniques described here results in a program that can play credible chess (or other games). Let us assume we have implemented an evaluation function for chess, a reasonable cutoff test with a quiescence search, and a large transposition table. Let us also assume that, after months of tedious bit-bashing, we can generate and evaluate around a million nodes per second on the latest PC, allowing us to search roughly 200 million nodes per move under standard time controls (three minutes per move). The branching factor for chess is about 35, on average, and \(35^{5}\) is about 50 million, so if we used minimax search, we could look ahead only about five plies. Though not incompetent, such a program can be fooled easily by an average human chess player, who can occasionally plan six or eight plies ahead. With alpha-beta search we get to about 10 plies, which results in an expert level of play. Section 5.8 describes additional pruning techniques that can extend the effective search depth to roughly 14 plies. To reach grandmaster status we would need an extensively tuned evaluation function and a large database of optimal opening and endgame moves.