Multi-game Playing

One of the ultimate goals of AI/CI research in games is development of a universal playing agent, able to play virtually any game as long as it knows its rules. Realization of this task requires designing general-purpose learning and reasoning methods that abstract from particular games. A potential variety of possible games to be played (theoretically there are infinite number of them) with different equipment (boards, card decks, various types of moving pieces, different goals, etc.) makes the task - in its general form - extremely demanding. Therefore, on a general level the goal is yet far from being accomplished, however, some steps in this direction have already been successfully taken.

Multi-game playing became a hot topic in the beginning of 1990s when some renowned methods and systems originated. In order to make the problem tractable, these first attempts were often restricted to a certain class of games, usually two-player, perfect-information, zero-sum, deterministic ones.

The majority of approaches developed in this area rely on symbolic, logical game representations, rooted in the mainstream of traditional AI. Relatively fewer examples can be found within the CI-related methods. Selected attempts, which originated in both AI or CI areas are briefly summarized in this chapter.

14.1 SAL

One of the first widely-known CI-based universal learning agents was Michael Gherrity’s SAL (Search And Learning) system [131] capable of learning any two-player, perfect-information, deterministic board game. SAL consisted of a kernel that applied TD-learning combined with neural network’s backprop algorithm in order to learn the evaluation function for a given game. The kernel was game-independent and remained unchanged for different games. The rules of making valid moves for any particular game were represented by a game-specific module. The system used only 2-ply search supported by the consistency search method [21] [22] appropriately modified by Gherrity.
SAL generated two evaluation functions, one for each playing side which allowed learning nonsymmetric games or imposing asymmetry in symmetric games, if necessary. Both evaluation functions were represented as one-hidden-layer MLPs. The current game situation was characterized by some generally-defined binary features (applicable to a wide spectrum of games), which included positional features - based on the board position (e.g. a type of a piece on each square, the number of pieces of each type, etc.), non-positional features - based on the last move made (the type of a piece moved, the type of a piece captured), and rule-based features (e.g. pieces potentially lost, squares potentially lost, possible win of a game, etc.). Since all features were binary, the number of them, especially in the case of varied pieces’ types, might be quite significant.

The size of the input layer depended on the number of game features generated for the game. The size of the hidden layer was arbitrarily chosen as being equal to 10% of the input size. Each move made during the game represented one training example. The target value for the output represented the evaluation of the next board position and was calculated with the TD(λ) method. Neural networks were trained with off-line backpropagation algorithm, i.e. after the game was completed.

SAL’s playing rate was strongly hindered by slow learning. For example, it took the program 20 000 games to learn to play tic-tac-toe. In the case of connect-4, SAL required 100 000 games to achieve approximately 80% winning rate over the training program (the details about the training program’s strength are not available), using 221 game features.

In a more serious attempt to learn how to play chess, SAL played 4200 games against GNU Chess, drawing 8 of them and losing the remaining ones. During the experiment GNU Chess was set to make a move within one second, which was approximately equivalent to 1500 – 1600 ELO rating. SAL was using 1031 input features and searched to 4-ply depth on average.

The analysis of games played against GNU Chess showed that SAL’s learning process had been extremely slow, but on the other hand a stable progress had been observed from the initial random play towards a more organized way of playing. An indication of this performance increase was the average length of games played by SAL (before being mated), which steadily increased from about 15 moves in the initial phase, to about 30 on average at the end of the experiment.

The question whether SAL would be capable of learning how to play chess (or another complicated game) at decent level within a reasonable amount of time remains open, but the answer is likely to be negative.

### 14.2 Hoyle

Another interesting approach to game-independent learning was represented by Hoyle system [93, 94, 97, 98, 99] devised by Susan Epstein. Hoyle was able to learn any two-player, deterministic, perfect-information game defined