Problema salto del cavallo (hill climbing)

Data una scacchiera NxN ed un cavallo posizionato su una casella trovare una sequenza di mosse che consenta al cavallo di occupare tutte le caselle delle scacchiera ciascuna esattamente una volta ~~prima di ritornare alla posizione di partenza~~ (rendiamo il cammino aperto affinché l'euristica funzioni).
Si risolva il problema utilizzando un algoritmo di ricerca ~~in profondità~~ hill climbing. Come funzione euristica verrà usata la regola di Warnsdorff, che privilegia i cammini col minor numero di possibili successori.

Analizziamo innanzi tutto le strutture dati utilizzare per modellare il problema:
la scacchiera è di dimensione NxN ed ha le righe e le colonne numerate da 1 a N, di conseguenza una generica posizione del cavallo sulla scacchiera è stata rappresentata come una coppia di interi (n. riga, n. colonna): `(int * int)`. Allora una soluzione parziale sarà un cammino aciclico sul grafo avente come vertici le caselle della scacchiera e come archi le possibili mosse del cavallo da ogni casella (il grafo è stato rappresentato solo implicitamente dalla funzione che definisce gli archi uscenti da un nodo, senza l'uso di una struttura dati definita appositamente, cosa che era comunque possibile aggiungendo le righe `type 'a graph = Graph of ('a -> 'a list);;` per definire il tipo e cambiando la funzione salto_cavallo con `salto_cavallo (Graph s) inizio n`, e passando come parametro negli esempi il grafo dopo averlo opportunamente creato con `let g = Graph(successori n);;`) e sarà rappresentato da una lista di posizioni del cavallo, quindi una lista di coppie di interi: `(int * int) list`. Infine, la pila su cui si effettuano le chiamate ricorsive del metodo che contiene tutte le soluzioni parziali fino ad allora generate sarà quindi rappresentata con una lista di liste di coppie di interi: `(int * int) list list`.
Analizziamo ora l'algoritmo: la funzione prende come argomento la dimensione N della scacchiera e la posizione iniziale del cavallo su di essa, e ci restituisce una lista che rappresenta le mosse con cui il cavallo tocca tutte le caselle della scacchiera una e una sola volta, senza però tornare obbligatoriamente al via (queste sono chiamate soluzioni aperte, e sono equivalenti ad un cammino hamiltoniano nel grafo che rappresenta la scacchiera e le mosse del cavallo). Ad ogni iterazione viene presa una soluzione parziale dalla pila: se essa tocca tutte le caselle della scacchiera allora è la soluzione che cercavo e la ritorno, altrimenti provo ad espanderla con la funzione estendi, che considera tutti i successori di un dato percorso e ritorna una lista di nuovi percorsi ottenuti da quello di partenza espandendolo di volta in volta con uno dei successori, a patto che esso non sia già stato toccato dalla soluzione di partenza. Le nuove soluzioni parziali vengono poi aggiunte in testa alla lista delle vecchie soluzioni parziali, ma prima vengono ordinate secondo la regola di Warnsdorff: vengono considerate prima le soluzioni che portano al minor numero possibile di nuove soluzioni parziali. Questo approccio Hill Climbing ci fornisce due vantaggi: il primo è quello di mantenere la pila delle soluzioni parziali la più piccola possibile, il secondo è che questa scelta ci consente di considerare prima soluzioni che percorrono prima i bordi e poi l'interno della scacchiera, così da evitare di lasciare zone irraggiungibili negli ultimi passi. Questa regola euristica ci consente di risolvere molte istante del problema, che di per se è un problema NP-completo, in tempo lineare rispetto alla dimensione della scacchiera.