Aufg 01)

Das Färbeproblem:
als Kodierung könnte ein Array verwendet werden, welches eine Abfolge von 0 und 1 speichert. Dabei bilden 3 Elemente eine Farbe als Binärzahl:
rot: 001
blau: 010
gelb: 011
grün: 100
lila: 101

Die Länder können anhand des Index ermittelt werden; die Länder repräsentierende Buchstaben haben eine feste Reihenfolge: (die Zahlen sind der Index)
0 1 2|3 4 5|6 7 8|9 10 11|12 13 14|15 16 17
A B C D E F
Das Array ist ein Individuum, also eine mögliche Kombination an Farben.

Vor dem Ausführen der Crossover Operation muss ein Wahrscheinlichkeitswert festgelegt werden, häufig 0,6. Also 60% wahrscheinlich, dass eine Rekombination stattfindet.

Crossover
1. 2x der Methodenaufruf selektion, um jeweils ein Individuum zu wählen; das sind unsere Eltern.
2. Generiere eine Zufallszahl zwischen 0 und 1.
3. Zufallszahl > 0,6 => Abbruch, nicht tauschen.
4. Zufallszahl < 0,6 => Methodenaufruf rekombiniere.

Methode selektion (auch zu Auswahl des Pools geeignet)
1. Berechne Fitness der gesamten Population
2. Berechne individuelle Auswahlwahrscheinlichkeit:
   p(gk) = individuelle Fitness/ Gesamtfitness
3. Bestimme die Intervallgrenzen des ‚Wahrscheinlichkeitsstrahls', die Wahrscheinlichkeiten der vorherigen Elemente inkl. mein Element aufsummiert.
4. Generiere eine Zufallszahl zw 0 und 1.
5. Suche das Individuum (Index) mit: Zufallszahl ist kleiner als rechte Intervallgrenze. (Wenn gleich: nächstes Individuum, da kein <=).
6. Gib das gewählte Individuum zurück

Methode rekombiniere:
1. Generiere eine Zufallszahl
2. Schneide beide Eltern an der Stelle der Zufallszahl
3. Speichere hintere Hälfte der Eltern a und b in Hilfsarrays Ha und Hb.
4. Überschreibe hinteren Teil von a mit Hb.
   Überschreibe hinteren Teil von b mit Ha.

Mutation:
Bei einer Binärcodierung bietet es sich an, für jedes „Gen", also jedes Bit zu prüfen, ob es „mutiert" wird. Dann wird es einfach geflippt, von 0 zu 1 und von 1 zu 0.
Es wird wieder ein Wahrscheinlichkeitswert festgelegt. Laut VL: von der Länge m abhängig.
1/18 = ca. 0,05

1. Für jedes Gen im Individuum:
    generiere eine Zufallszahl
        Zufallszahl < 0,05 => flippe Bit
            prüfe ob Zahl gültig ist; (zB 111 ist ungültig)
                bei ungültiger Zahl, wähle Zufallszahl
        Zufallszahl > 0,05 => tue nichts

Fitnessmethode:
hier habe ich mir zuerst die Bedingungen angesehen:
Die Farbe von:
A darf nicht gleich B|C sein
B darf nicht gleich A|C|D sein
C darf nicht gleich A|B|D sein
D darf nicht gleich C|B|E sein
E darf nicht gleich D sein
F darf jede Farbe haben; wird erst bei der Anzahl der Farben interessant

Für die Fitnessfunktion würde ich diese Bedingungen vergleichen. Sind sie erfüllt, ziehe ich einen Wert ab, sind sie nicht erfüllt, addiere ich einen Wert auf. Je höher der Wert, desto besser also. Ich habe die Werte einfach als +1/ -1 gewählt, da alle Bedingungen hier gleich viel Wert sind.
Zum Schluss würde ich noch die Anzahl an Farben überprüfen. Dazu würde ich die bisher kleinste Anzahl genutzter Farben speichern und wenn ein Individuum diesen Wert unterschreitet, dann bekommt das Individuum +5. (Und der Wert wird aktualisiert) Möglichst wenig Farben zu haben, ist ja das Ziel. (Bei Umsetzung habe ich mich auf Addieren beschränkt).

Die Funktion sähe etwa so aus:
vergleiche A mit B und C
vergleiche B mit C und D
Vergleiche C mit D
vergleiche D mit B und E
(die doppelten habe ich nicht mit aufgeführt)

Bestimme Anzahl genutzter Farben

Gebe einen Int-Wert zurück. Das ist die Fitness eines Individuums.

Damenproblem
Kodierung als Vektor, aber mit binärzahlen von 1-8. Ähnlich wie bei obigem Problem, haben die Spalten des Brettes eine feste Reihenfolge:
[001 010 011 100.... 111]
 a. b. c. d ....g.
Die binäre Darstellung, weil dies ein genetischer Algorithmus ist.

Crossover
Genau wie im oberen Problem.

Mutation
bleibt wie oben; jedes Gen wird durchgegangen und das Bit mit einer gewissen Wahrscheinlichkeit geflippt. Nur die Anzahl der genutzten Bits muss angepasst werden.

Fitnessfunktion

Die Bewertung setzt sich aus der Anzahl der Konflikte zusammen. Deshalb würde ich in der Funktion die Zeile und die Diagonalen ausgehend von meiner aktuellen Dame prüfen. Die Spalten können keine zwei Damen haben, da für jede Spalte eine Dame generiert wird.

Das sähe etwa so aus:
Gibt es eine oder mehrere Damen in der Zeile meiner Dame?
Ja=> + Anzahl Damen
Gibt es eine oder mehrere Damen in den Diagonalen meiner Dame?
Ja=> + Anzahl Damen

Die Zeile kann einfach überprüft werden: Der Wert im Array ist gleich.

Die Diagonalen sind schwieriger. Mein erster Ansatz war, jedes Feld abzugehen so +1/-1 zur Seite, +1/-1 nach oben oder unten, dann alles +2.... Das wird aber sehr schnell viel Code. Deshalb habe ich nach einer rechnerischen Lösung gesucht und bin schnell auf Vektorrechnung gestoßen. Wenn man nach links/ rechts genauso weit geht, wie nach oben, bzw. unten, sind die Differenzen von x2-x1 und y2-y1 gleich.

Je weiter das Ergebnis vom Anfangswert 0 ist, desto schlechter das Ergebnis. (In der Implementierung habe ich gemerkt, dass ich lieber mit positiven Werten arbeite. Deshalb habe ich eine hohe positive Zahl als Ausgangswert genommen und die Konflikte abgezogen.)

Um die Probleme mit Simulated Annealing lösen zu können, bräuchte es noch eine Temperatur, die schrittweise abgekühlt wird. Für die Temperatur bräuchte man einen Plan, einen Schedule, um die Temperatur zu senken, sie aber nicht zu schnell zu senken.
Dann müsste man eine Methode haben, die die Wahrscheinlichkeit, ob ein schlechterer Knoten bzw. Zustand gewählt wird, anhand der Temperatur, der Bewertung und der e-Funktion berechnet.