12.ÜNİTE SEZGİSEL ARAMA

12.4 TEPE TIRMANIŞI
12.4.1 TEPE TIRMANIŞI ÖRNEĞİ
12.4.2 KAPALI ŞÖVALYE TURU
12.4.3 VEZİRLERİN N- PROBLEMİ

ÖZGE SAYINBAŞ
2023481057

12.4 TEPE TIRMANIŞI

Derinlik öncelikli arama, çözümü körü körüne aradığı için pratik değildir. Eğer arama gerçekten körse, bazen şanslı oluruz ve hızlıca çözümü buluruz, diğer zamanlarda ise özellikle sonsuz dallanmalar olduğunda, arama alanının büyüklüğüne bağlı olarak hiçbir zaman çözüm bulamayabiliriz.
Eğer hedefin nerede olduğu hakkında biraz daha bilgi sahibi olsaydık, derinlik öncelikli arama algoritmasını iyileştirebilirdik. Bir dağın zirvesine ulaşmaya çalıştığınızı düşünün. Dağın zirvesini görebildiğimizden dolayı oraya ulaşmak için izlememiz gereken genel yönü biliyoruz. Tepeye tırmanmak istiyoruz. İşte bu algoritmanın adı buradan gelir.

Dağ tırmanışı yapan herkes, bazen dağa çıkan bir yolun bir çıkmaza götürdüğünü bilir. Bazen bir rota gibi görünen şey sadece yakınlarda daha küçük bir zirveye yol açabilir. Bu yanıltıcı zirvelere yerelleştirilmiş maksimum denir ve tepe tırmanışı, bir yerelleştirilmiş maksimum bulma ve bunun aranan genel hedef olduğunu düşünme riski taşır. Şekil 12.3, aramaya tepe tırmanışı uygulanmış aynı labirenti göstermektedir. Tırmanmak için yardımcı olması için bir sezgisel yöntem uygularız. Bir labirenti ararken, labirentin çıkış noktasını biliyorsak, Manhattan mesafesini bizi hedefe yönlendirmek için bir sezgisel olarak kullanabiliriz. Çözüme götürecek yolun uzunluğunu bilmiyoruz çünkü labirentin tüm detaylarını bilmiyoruz, ama labirentin neresinden ne kadar uzakta olduğunu tahmin edebiliriz. Hedefin yerini ve mevcut konumumuzu biliyorsak hedefe varmışızdır

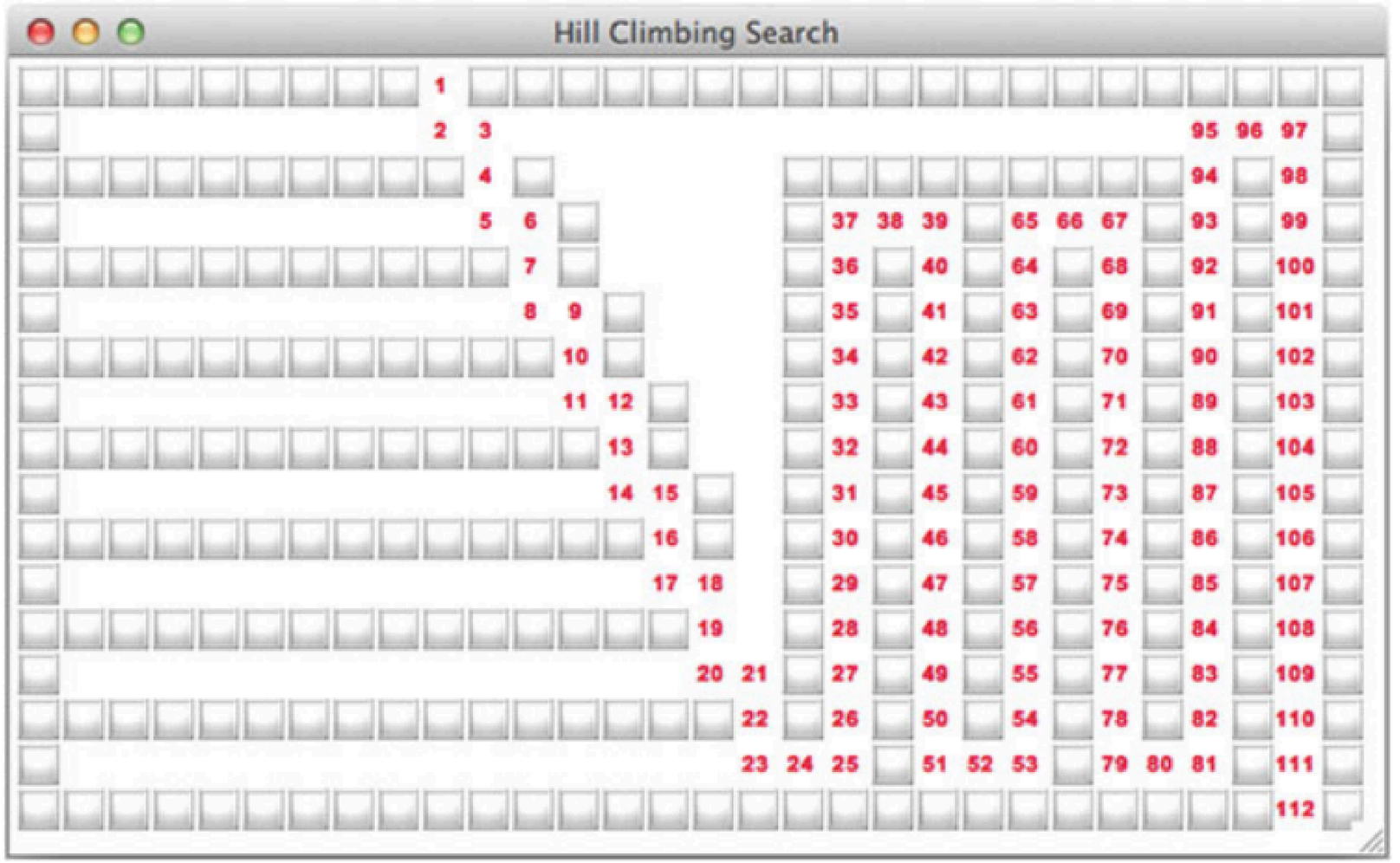

Fig. 12.3 Hill Climbing Search of a Maze

Manhattan mesafesi, bir labirent veya harita üzerindeki herhangi iki konumu ayıran satır ve sütun sayısının bir ölçüsüdür. Şekil 12.3'te, başlangıçtan hedefe olan Manhattan mesafesi 36'dır. Bir satır aşağıya inmemiz, ardından 20 sütun sağa gitmemiz ve 15 satır aşağı inmemiz gerekir. Bu mesafe, Manhattan mesafesi olarak adlandırılır çünkü bu, Manhattan'daki binalar arasında veya herhangi bir şehirdeki şehir bloklarında yürümek gibidir. Manhattan mesafesi ya tam ya da hedefe olan toplam mesafenin bir alt tahmini olacaktır. Şekil 12.3'te bu tam bir tahmin, ancak genel olarak doğrudan bir rota mümkün olmayabilir, bu durumda Manhattan mesafesi bir alt tahmin olacaktır. Bu önemlidir çünkü mesafeyi fazla tahmin etmek, tepe tırmanışının tekrar derinlik öncelikli arama gibi çalışmasına neden olacaktır. Sezgisel, algoritmanın performansını etkilemeyecektir. Örneğin, eğer kolay olanı alırsak yaklaşım ve mesafemizin 100 olduğunu söyleseydik, tepe tırmanışı gerçekleşmeyecektir Şekil 12.3'teki örnek, mümkünse önce aşağı gitmeyi tercih ettiğini gösterir. Sonra sağa gider. Hedef konumu bilinir ve minimum Manhattan mesafesi keşfedilecek seçenekleri düzenler. Başka bir seçenek yoksa sol veya yukarı gitmek bir seçenek değildir. Bu nedenle, algoritma, bu yolu izleyerek 25. adıma kadar aşağı ve sağa doğru ilerler, bu noktada bu yolda başka bir seçeneği olmadığı için yukarı gitmek zorundadır.

Tepe tırmanışı, bir çıkmaza ulaşana kadar bir yoldan vazgeçmeyeceği için derinlik öncelikli arama gibi davranır. Tepe tırmanışı Şekil 12.3'te en iyi çözümü bulmaz, ancak bu durumda genişlik öncelikli veya derinlik öncelikli aramadan çok daha az konumu inceler. Tepe tırmanışının avantajları ve dezavantajları şöyledir.
*Hedefin konumu arama başlamadan önce bilinmelidir.

*Hedefe olan yolun uzunluğunu alt tahmin eden veya tam bir uzunluk sağlayan bir sezgisel olmalıdır. ?
Sezgisel ne kadar iyi olursa, tepe tırmanışı arama algoritması o kadar iyi olur.

*Tepe tırmanışı, büyük arama alanlarında bile iyi performans gösterebilir.
*Sezgisel arama dallanmalarını engelleyebiliyorsa,
tepe tırmanışı sonsuz arama dallanmalarıyla başa çıkabilir.

*Tepe tırmanışı yerel maksimumlardan veya zirvelerden etkilenebilir.

*Tepe tırmanışı, genişlik öncelikli arama gibi optimal bir çözüm bulmayabilir.

Tepe tırmanışını uygulamak için her adımdaki alternatif seçenekler, yığına yerleştirilmeden önce sezgisel bir sıraya göre sıralanır. Aksi takdirde, kod, derinlik öncelikli aramanınkiyle tam olarak aynıdır.

12.4.2 KAPALI AT TURU

Tepe tırmanışı, kapalı at turu problemi çözümünde kullanılabilir. Bu problemi çözmek, bir atı satranç oyununda bir satranç tahtası (veya herhangi bir boyutta tahta) etrafında hareket ettirmeyi içerir. At, satranç tahtasında bir L oluşturarak iki kare ilerleyip bir kare dik yönde hareket etmelidir. Kapalı at turu problemi, başlangıç ve bitiş noktası aynı olan ve başlangıç ve bitiş noktası dışında hiçbir karenin iki kez ziyaret edilmediği bir dizi yasal at hareketiyle tahtadaki her konumu ziyaret eden bir yol bulmaktır. Tahtada bir yol bulmak istediğimizden, çözüm başlangıçtan bitişe bir yol olarak temsil edilebilir. Yol üzerindeki her düğüm, tahtadaki bir hamledir. Bir hamle, tahtada olması ve zaten yolun içinde olmaması durumunda geçerlidir. Bu şekilde, tahta açıkça oluşturulmamış olur.

Bir at için olası hareketleri oluşturmak, tahtanın kenarlarıyla başa çıkmak için kod yazmaya çalışırsanız oldukça karmaşık olabilir. Genel olarak, komşu düğümlerin oluşturulması gerektiğinde ve sınırlar üzerinde özel durumlar ortaya çıktığında, geçerli olmayan hareketlerle birlikte geçerli hareketlerin bir kümesini oluşturmak çok daha kolaydır. Atın etrafında hareket etme durumunda, genel olarak sekiz olası hareket vardır. Tüm olası hareketler oluşturulduktan sonra, geçersiz hareketler açıktır ve filtrelenir. Bu teknik kullanılarak, sınırlar bir kez ele alındığında her bir ayrı olası hareket yerine tutarlı bir şekilde ele alınır. Kod çok daha temiz ve mantık çok daha anlaşılır hale gelir.

Şekil 12.4, 12×12 bir tahta için kapalı at turu problemine bir çözüm sunar. Tur, hesaplanırken turun nerede başladığını ve bittiğini görebileceğiniz şekilde alt sol köşeden başlar. Turun bir sonraki konumun seçeneklerini sıralamak için bir sezgisel kullanarak bulmak birkaç dakika sürer. En az kısıtlanmış bir sonraki seçenek, en fazla kısıtlamaya sahip olan seçenektir. Bu şekilde, bir sonraki hamle en fazla seçeneğe sahip olduğundan genellikle çıkmazlara götüren yollara bakmamayı sağlar. Başka bir deyişle, en fazla seçeneğe sahip olan yerlere yakın kalmayı ve tahtanın ortasında sıkışıp kalmamayı tercih eder. Bu sezgisel yöntem mükemmel değildir ve çözümü bulmak için hala bazı geri izleme gereklidir. Yine de, bu sezgisel olmadan 12x12 tahta için problemi makul bir sürede çözme umudu olmazdı. Aslında, 8x8 çözümü basit bir derinlik öncelikli arama ile makul bir sürede bulunamaz, her adımda doğru yönde arama şansınız olmadıkça. Sezgisel ve tepe tırmanışı uygulandığında, 8x8 çözümü sadece birkaç saniyede bulunabilir.

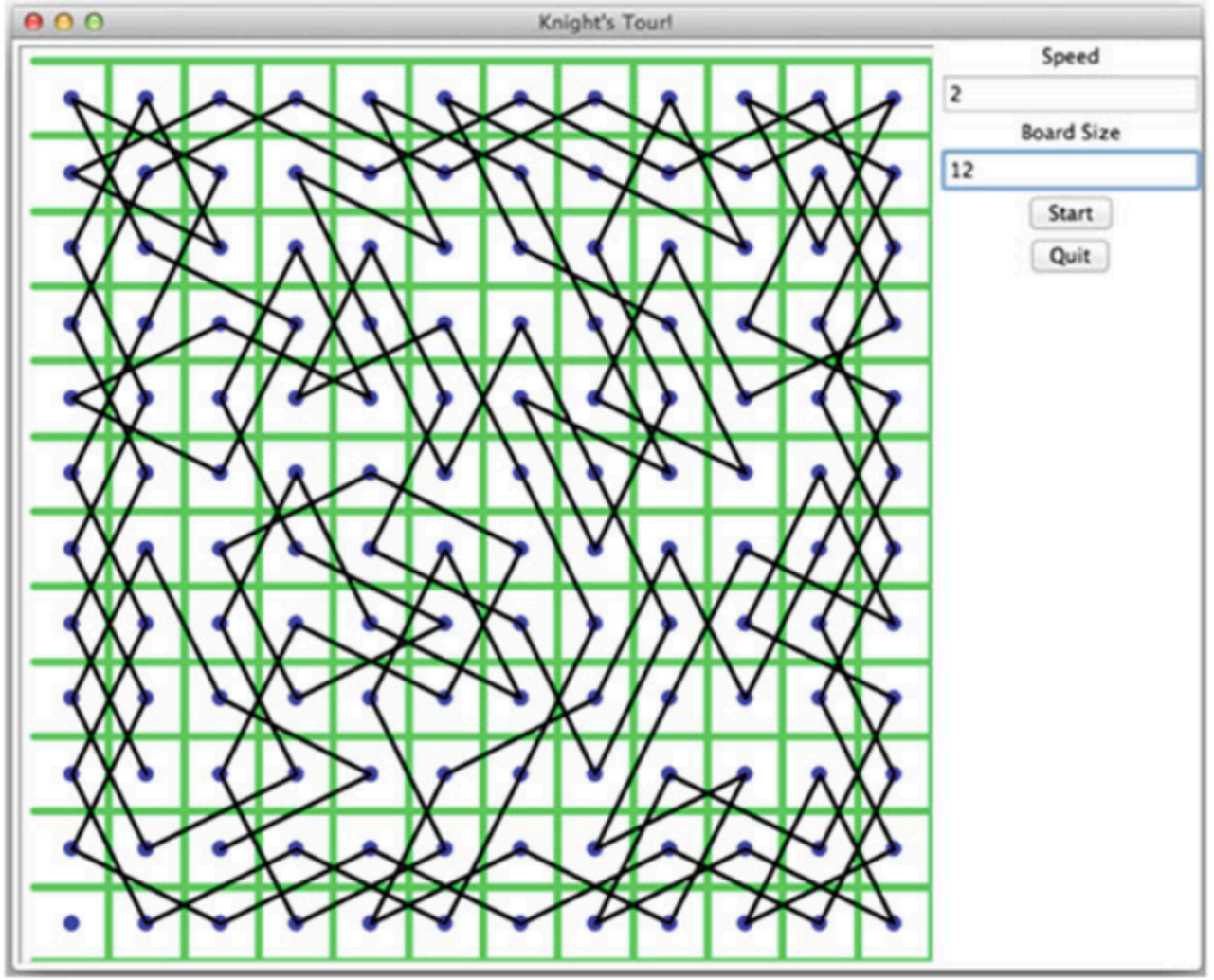

Fig. 12.4 A Closed 12 × 12 Knight's Tour

12.4.3 VEZİRLERİN N-PROBLEMİ

Vezirlerin N- Problemi, N vezirin bir N×N satranç tahtasına yerleştirilmesi gerektiğinde, hiçbir iki vezirin aynı sütun, satır veya köşegen üzerinde olmadığı şekilde çözülmelidir. Bu problemi derinlik öncelikli arama kullanarak çözmeye çalışmaz. Arama alanı çok büyüktür ve basit kaba kuvvet kullanarak bir çözüm bulmak için çok şanslı olmanız gerekir. Vezirlerin N- Problemi benzersiz bir özelliği vardır ; bir vezir yerleştirildiğinde tahta üzerinde, yerleştirildiği satır, sütun veya köşegenlerdeki diğer tüm konumlar artık gelecekteki hamleler için olası adaylar değildir. Bu olası hamleleri kaldırma mevcut konumlar listesinde ileri kontrol olarak adlandırılır. Bu ileri kontrol her adımda arama uzayının boyutunu azaltır.

Bir vezirin yerleştirileceği bir sonraki satırın seçimi, vezirlerin N- problemi için başka bir benzersiz özelliktir. Rastgele bir satırın seçilmesi veya sadece sıralı satırlar dizisinden bir sonraki satırın seçilmesi çözüm bulmayıkolaylaştırmaz. Bu nedenle, çözüm sadece bir sonraki veziri hangi sütuna yerleştireceğimizdir.
İleri kontrolde yardımcı olmak için, tahta bir tuple olarak temsil edilebilir: (vezir konumları, kullanılabilir konumlar). Tuple'ın ilk öğesi yerleştirilmiş vezirlerin listesi iken, tuple'ın ikinci öğesi tahtadaki kullanılabilir konumların listesidir. İleri kontrol, bir sonraki satır için kullanılabilir konumlardan birini seçebilir. Bu noktada, tuple'ın ikinci bölümünde, bir sonraki vezir yerleştirmesi seçimiyle çakışan tüm konumlar ortadan kaldırılabilir. Böylece ileri kontrol, bir vezirin yerleştirilmesi için bir seçim yapıldığında artık uygun olmayan tüm olası konumları ortadan kaldırır.

Vezirlerin N- Problemi' nin çözümünde tepe tırmanışı kısmı, bir vezirin hangi sütuna yerleştirileceği seçildiğinde devreye girer. Seçilen sütun, gelecekteki seçenekleri en az kısıtlayan sütundur. At Turu gibi, Vezirlerin N-Problemi, bir sonraki seçim yapıldığında daha fazla seçenek bıraktığı zaman faydalanır. Bu sezgisel kullanılarak, ileri kontrol ve bir sonraki vezirin yerleştirileceği satırın basit seçimi ile, 25-Vezir problemi makul bir sürede çözülebilir. Bir çözüm Şekil 12.5'te gösterilmiştir. Gözden geçirilecek olursa, tepe tırmanışının uygulanması, her adımdaki alternatif seçeneklerin yığına yerleştirilmeden önce sezgisel bir şekilde sıralanmasını gerektirir. Aksi takdirde, kod, derinlik öncelikli arama ile aynıdır. Bazı durumlarda, At Turu ve Vezirlerin N- Problemi gibi, herhangi bir çözüm optimal bir çözümdür. Ancak, yukarıda bir labirent ararken belirtildiği gibi, tepe tırmanışı her zaman optimal bir çözüm bulmaz.

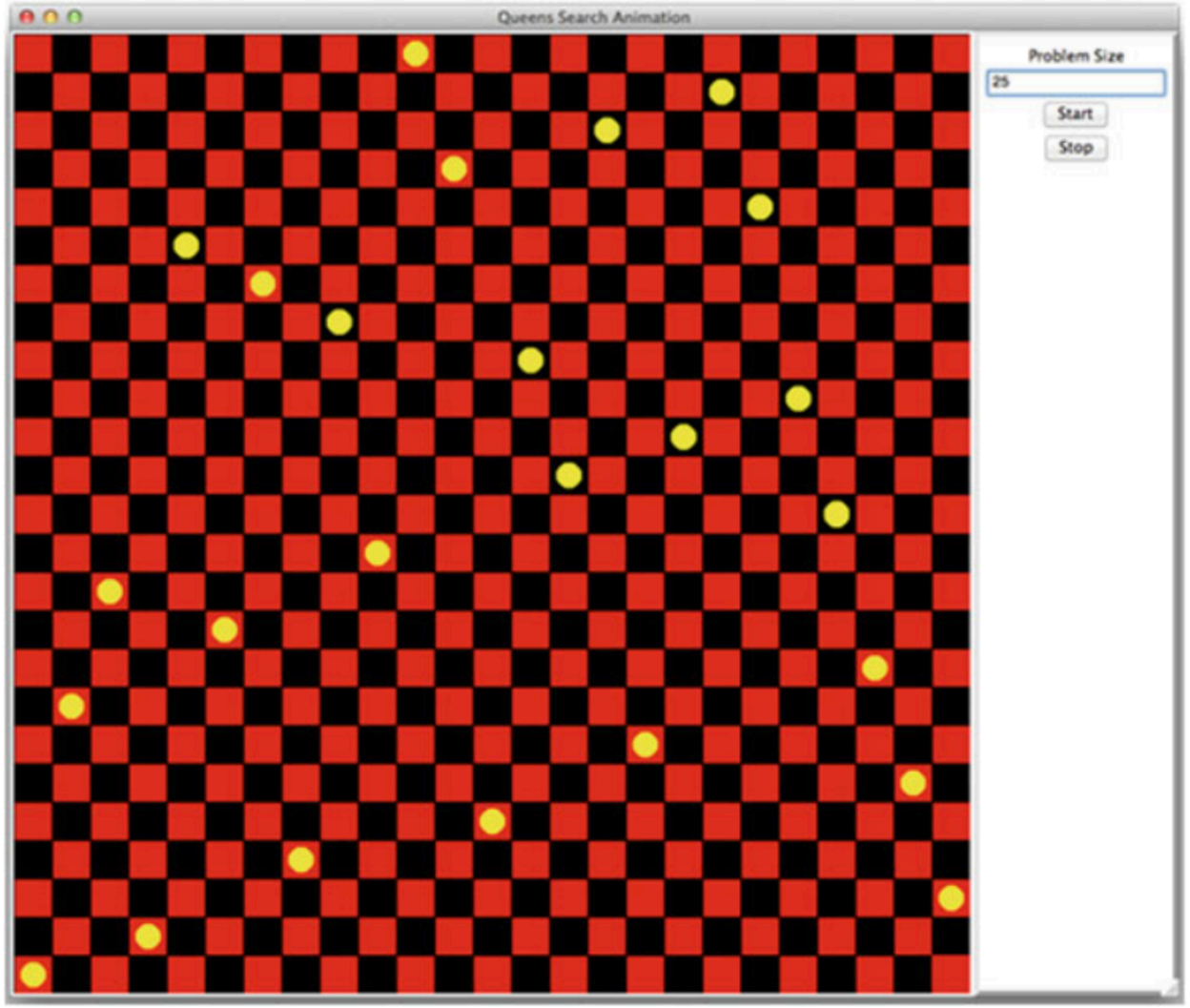

Fig. 12.5 A 25-Queens Solution

7. SORU
İLERİ KONTROL NEDİR VE BİR SORUNU ÇÖZMEYE NASIL YARDIMCI OLUR?

İleri kontrol, bir sistemin gelecekteki durumlarını tahmin etmek ve bu tahminlere dayanarak uygun kontrol eylemlerini önceden belirlemek için kullanılan bir kontrol stratejisidir İleri kontrol, gelecekteki durumları öngörme ve bu öngörülere göre sistem davranışını optimize etme kabiliyeti sayesinde, çeşitli mühendislik ve endüstri alanlarında önemli faydalar sağlar. Bu yaklaşım, sistemlerin daha verimli, güvenilir ve kararlı bir şekilde çalışmasına yardımcı olur.