Harmony Search Algorithm for solving Constraint Satisfaction Problems

Ayad M. Turky - Sadir A. Fadhil
College of Computer, University of Anbar, Alanbar, Iraq.
Ayad_b2006@yahoo.com - Sadir_it@yahoo.com

Abstract:

In this paper we present a harmony search (HS) algorithm for solving constraint satisfaction and optimization problems. The Harmony search algorithm is an evolutionary algorithm which mimics musicians' behaviors such as random play, memory-based play, and pitch-adjusted play when they perform improvisation. The major thrust of this algorithm lies in its ability to integrate exploitation and exploration in a parallel optimization environment. This algorithm results are compared with other algorithms using the well known n-queens problem.

Keywords: Constraint Satisfaction Problems (CSP), Harmony search, N-queens problem.

المستخلص :

خوارزمية البحث ألتناغمي لحل المشاكل الصعبة آو المقيدة الإرضاء

قدمنا في هذا البحث خوارزمية البحث ألتناغمي لحل المشاكل الصعبة آو المقيدة الإرضاء. وهي خوارزمية تطورية تقلد سلوك الموسيقيين من ناحية العشوائية في الأداء المعتمد على الذاكرة في عملية التغيير في النغمات عند القيام بالارتجال في الأداء. الهدف الرئيسي لهذه الخوارزمية يكمن في قدرتها في دمج الاستغلال والاستكشاف في بيئة التحسين.

تم تطبيق هذه الخوارزمية لحل مشكلة (N-queens) ، النتائج التي تم التوصل إليها تم مقارنتها مع نتائج خوارزميات أخرى .

1 Introduction

The field of metaheuristics for the application to constraint satisfaction and optimization problems is a rapidly growing field of research. A finite constraint satisfaction problem (CSP) consists of a set of variables associated with finite domains and a set of constraints restricting the values that the variables can simultaneously take. A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied [1] and [2].
There is a class of problems in the fields of artificial intelligence, network, database, engineering and other areas of computer science can be viewed as special cases of constraint satisfaction problems (CSP), including image processing, natural language parsing, routing, circuit design, scheduling and more, where the search space is very large and the search path does not need to be remembered. These problems cannot be solved by a classical search algorithm or linear programming such as a breadth-first search, an algorithm guaranteed to find the solution, because the search space is not computable in polynomial time.

Metaheuristic algorithms came to the fore to solve the constraint satisfaction problems (CSP). Commonly, metaheuristic algorithms are divided into two types. The first type is a local based algorithm which starts with one solution and tries to satisfy the constraints iteratively based on a fitness function until a global optimal solution is reached. The main drawback of those local based algorithms is that they may get stuck in the local optimal solution in which the current solution is considered the best notwithstanding the fact that there are other solutions in different areas of the search space with a better quality. The

main cause for the local optimal problem is that local based algorithms focus on exploitation rather than exploration which means that the local based algorithms move in one direction without performing a wider scan of the search space.

The second type of metaheuristic algorithms is a population based algorithm which starts with many different solutions and refines them in a parallel optimization environment until a global optimal solution is reached. Unfortunately, the quality of the solution produced by *population based algorithms is inferior to local based algorithms mainly* due to the fact that the population based algorithms normally experience premature convergence, the main reason for this problem in population based algorithms are that they are more concerned with exploration rather than exploitation. That is, the population based *algorithms scan the solutions in the whole search space without* rigorous concentration on those current solutions in addition to other drawbacks such as the need for more time.

From the above, it's clearly that the best algorithm to solve constraint *satisfaction problems (CSP) that have provide a good balance among* local based algorithms and population based algorithms to reach a suitable balance between exploration (global improvement) and exploitation (local improvement).

This paper investigates the effectiveness of harmony Search (HS) in searching for good quality solutions for N-Queens problem. The purpose of studying the N-Queens problem is to provide a useful indicator of the potential solutions for solving different constraint

satisfaction problems [3]. The rest of the paper is organized as follows: Section II reviews the N-queens problem. Section III describes the application and implementation of Harmony Search (HS) to this problem, Section IV discusses the experimental results and Section V concludes the paper.

2 N-queens problem

The N-Queens problem was originally introduced in 1848 by M. Bezzel [4]. The goal of this problem is to place n queens on a n × n chessboard so that no queen is under a direct attack from any other one. This problem can be made more difficult to solve by setting some prohibited fields on chessboard (N-queens problem with holes). None of the queens can be placed on these fields.

At first, we need to model the N-queens problem as a CSP problem:

- Variables $\{Q_1, Q_2, Q_3 \ldots Q_n\}$ represent the queens,
- Domains Q □ {1,2,3…n} □*i* ▯ {1,2,3…n},

Where equality $Q\ i = j$ determines the *i*-th queen is placed on the *i*-th column and *j*-th row (note, that each queen strictly determines the column where it is placed),

- Constraints
  - ✓ condition for rows
    $$Q_i \neq Q_j \quad \forall i, j \in \{1,2,\cdots n\},\ i \neq j$$
  - ✓ diagonals
    $$|Q_i - Q_j| \neq |i - j| \quad \forall i, j \in \{1,2,\cdots n\}, i \neq j$$
  - ✓ prohibited fields (holes)
    $$Q_i \neq j \quad \forall (i, j) \in H$$

Where H is the list of holes – pairs (*column*, *row*).
Prohibited fields constraint can be directly propagated to the queens' domains.

Figure 1 shows an example of non-conflicting situations of 4-Queens problem

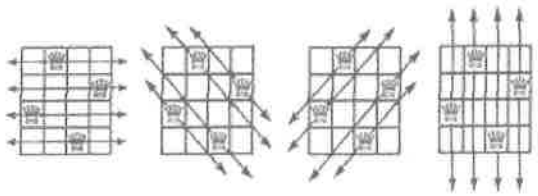

Fig.1. Non-conflicting Placement of 4-Queens problem

3 Harmony Search Algorithm Implementation

Harmony Search (HS) was first developed by Zong Woo Geem et al. in 2001 [5], though it is a relatively new metaheuristic algorithm, its effectiveness and advantages have been demonstrated in various applications. Since its first appearance in 2001, it has been applied to solve many optimization problems including function optimization, engineering optimization [6], water distribution networks [7],[8] and [9], groundwater modeling, energy-saving dispatch, truss design, vehicle routing, and others.
Harmony search (HS) algorithm was recently developed in an analogy with music improvisation process where music players improvise the pitches of their instruments to obtain better harmony [5] , [10].

The steps in the procedure of harmony search are as follows:
Step 1:

For the first step of the HS algorithm, the HM matrix is filled with as many randomly generated solution vectors as the HMS as follows:

$$HM = \begin{pmatrix} x_{11}^{1} & x_{12}^{1} & \dots & x_{1n}^{1} \\ x_{21}^{2} & x_{22}^{2} & \dots & x_{2n}^{2} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ x_{n1}^{HMS} & x_{n2}^{HMS} & \dots & x_{nn}^{HMS} \end{pmatrix}$$

Where n is number of queens, i and j represents the row and column number, respectively in n^{th} vector stored in HM;

Next step, a new harmony is improvised using one of the following three mechanisms: memory consideration, random selection, and pitch adjustment. As follows :

$$HM^{NEW} = \begin{pmatrix} x_{11}^{NEW} & x_{12}^{NEW} & \dots & x_{1n}^{NEW} \\ x_{21}^{NEW} & x_{22}^{NEW} & \dots & x_{2n}^{NEW} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ x_{n1}^{NEW} & x_{n2}^{NEW} & \dots & x_{nn}^{NEW} \end{pmatrix}$$

Step 2:

Memory consideration: *Operator selects the queens positions of New Harmony solution based on solutions stored in HM with probability harmony memory considering rate (HMCR).*

$$x_{ij}^{NEW} \leftarrow x_{ij}, \quad x_{ij} \in \{x_{ij}^{1}, x_{ij}^{2}, \ldots, x_{ij}^{HMS}\}$$

W.P. (HMCR)

Step 3:

Random Selection: (1-HMCR) Queens that are not placed depending on memory consideration will be assigned to the New Harmony solution randomly from the available chessboard range. In fact, this process is very important to diversify the New Harmony solution.

$$x_{ij}^{NEW} \leftarrow x_{ij}, \quad x_{ij} \in \{1, 2, \ldots, N\}$$

W.P. (1-HMCR)

Step 4:

Pitch adjustment: After the New Harmony solution is generated, the pitch adjusting operator will be applied to the New Harmony solution with a probability Pitch Adjusting Rate (PAR). This operator will examine all queens that are placed out of harmony considerations.

Once one pitch is obtained in memory consideration rather than random selection, the obtained value may further move to neighboring values with a probability of HMCR × PAR while the original value obtained in memory consideration does not move with a probability of HMCR × (1-PAR). PAR ($0 \leq PAR \leq 1$) stands for pitch adjusting rate. The obtained the queen that move to neighboring place with a probability (PAR × HMCR) value where $0 \leq PAR \leq 1$. The other queen with a probability rate ((1-PAR) × HMCR) Rare not changed their place.

$$X_{i,j}^{NEW} = \begin{cases} Yes & w.p. \quad (HMCR \times PAR) \\ No & w.p. \quad (HMCR \times (1-PAR)) \end{cases}$$

If the selected queen will move to a valid position or swap with another course with the probability of PAR□HMCR. Here, pitch adjusting operator works similar to neighborhood structures of local based algorithms that is concerned with the exploitation of the New Harmony solution while memory consideration operator is concerned with exploration.

In case the new harmony vector HM^{NEW} is better than the worst harmony in the HM in terms of objective function value, $Z(HM^{NEW})$, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.

If the stopping criterion (maximum number of improvisations) is satisfied, computation is terminated. Otherwise, another new harmony is improvised by considering one of three mechanisms.

4 Experimental Result

In this section we present the efficiency of the described harmony search algorithm on the N-queens problem. The algorithm was programmed using JAVA and the simulations were performed on the PC AMD Athlon with a 1.92 GHz processor and 512 RAM running Windows XP 2002.

Table 1 show the parameters used in the harmony search algorithm after preliminary experiments.

Parameter	Value
HMS	50
HMCR	0.7
PAR	0.3

Table 1. Parameter Setting

The results are presented in Table 2. This table also contains the comparison of our best results with those appeared in the literature, e.g. [11] (**our best results are shown in bold**).

Number of queens	Our best Results [HS]	Number of iterations [HS]	Muller[11]	Number of iterations [11]
100	10 ms	100	8 ms	120
500	**100** ms	285	126 ms	528
1000	**267** ms	500	558 ms	1017
2000	**1928** ms	1000	2375 ms	2022
5000	**13132** ms	1500	16031 ms	5026
6000	17102 ms	10000	---	---
7000	19562 ms	12000	---	---

Table 2. Result

Our Harmony search [HS] algorithm approach obtained better results compared with other algorithms applied for solving N-Queens problem

as shown in Table 2. Consequently, we may conclude that our approach is generally able to produce good results when compared against other algorithms.

5 Conclusion

A recently-developed music phenomenon-inspired algorithm, HS was introduced for solving the N-Queens problem. As it has been shown, the results show that the N-Queens problem can be successfully solved using HS algorithm and produces better results than several others in the previous literature. Although the N-Queens problem does not have much practical use, it represents a large class of NP problems that cannot be solved in a reasonable amount of time using deterministic methods.

From these results, the HS algorithm appears to have a potential to be successfully applied to constraint satisfaction problems (CSP).

References

1.R. Barták. On-line Guide to Constraint Programming, http://kti.mff.cuni.cz/~bartak/constraints, 1998.

2. P. Hentenryck. Constraint Satisfaction in Logic Programming. MIT Press, 1998.

3. J. Gu, Constraint-Based Search, New York: Cambridge University Press, 1992.

4.M. Bezzel, Proposal of 8-queens problem, Berliner Schachzeitung 3 (1848) 363. Submitted under the author name "Schachfreund".

5. Geem ZW, Kim JH and Loganathan GV (2001) A new heuristic optimization algorithm:

Harmony search. Simulation, 76:60-68.

6. Lee KS and Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering
optimization: harmony search theory and practice. Comput. Methods Appl. Mech. Engrg.,
194:3902-3933.
7. Geem ZW (2006) Optimal cost design of water distribution networks using harmony
search. Engineering Optimization 38:259-280.
8. Geem, Z.W., Lee, K.S., Park, Y.: Application of Harmony Search to Vehicle Routing.
American Journal of Applied Sciences. 2(12), 1552–1557 (2005).
9. Geem, Z.W., Choi, J.–Y.: Music Composition Using Harmony Search Algorithm. In:
Lecture Notes in Computer Science, vol. 4448, pp. 593–600 (2007).
10. Ryu, S., Duggal, A.S., Heyl, C.N., Geem, Z.W.: Mooring Cost Optimization via Harmony Search. In: Proceedings of the 26th International Conference on Offshore Mechanics and Arctic Engineering (OMAE 2007), ASME. CD-ROM (2007).
11.T. Müller, Interactive Heuristic Search Algorithm In Proceedings of the CP'02Conference - Doctoral Programme, Ithaca, September 2002