Computers & Industrial Engineering 60 (2011) 310–319

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Self Controlling Tabu Search algorithm for the Quadratic Assignment Problem ☆

Nilgun Fescioglu-Unver [a,*], Mieczyslaw M. Kokar [b]

[a] Department of Industrial Engineering, TOBB University of Economics and Technology, Ankara, Turkey
[b] Electrical and Computer Engineering Department, Northeastern University, Boston, MA, USA

ARTICLE INFO

Article history:
Received 10 November 2008
Received in revised form 24 November 2010
Accepted 24 November 2010
Available online 29 November 2010

Keywords:
Self-controlling software
Tabu search
Reactive search
Quadratic Assignment Problem

ABSTRACT

This paper addresses the application of the principles of feedback and self-controlling software to the tabu search algorithm. We introduce two new reaction strategies for the tabu search algorithm. The first strategy treats the tabu search algorithm as a target system to be controlled and uses a control-theoretic approach to adjust the algorithm parameters that affect search intensification. The second strategy is a flexible diversification strategy which can adjust the algorithm's parameters based on the search history. These two strategies, combined with tabu search, form the Self Controlling Tabu Search (SC-Tabu) algorithm. The algorithm is implemented and tested on the Quadratic Assignment Problem (QAP). The results show that the self-controlling features of the algorithm make it possible to achieve good performance on different types of QAP instances.

1. Introduction

Many real life combinatorial optimization problems are classified as NP-hard. Finding solutions to such problems within reasonable time limits, may not be possible (Garey & Johnson, 1979; Hertz, Taillard, & de Werra, 1997, cha 5). Heuristic algorithms often provide good solutions to optimization problems within reasonable time limits, however heuristic algorithm performance depends heavily upon the selected values for several parameters of the algorithms. A heuristic with a given set of parameter values that performs well on one type of problem may perform poorly on another type. The selection of appropriate values for search parameters is an active area of research (Adenso-Díaz & Laguna, 2006; Hutter, Hamadi, Hoos, & Leyton-Brown, 2006; Xu, Chiu, & Glover, 1998).

Two approaches to parameter selection have been described in the literature: parameter tuning and reactive search. Off-line parameter tuning methods use experimental design techniques and statistical analysis methods (Xu et al., 1998) to adjust algorithm parameters so that a search algorithm performs well on a given set of problems. Procedures combining experimental design and local search techniques are also used for this purpose (Adenso-Díaz & Laguna, 2006). Machine learning is often used to tune algorithm parameters automatically (Birattari, Stutzle, Paquete, & Varrentrapp, 2002; Hutter et al., 2006). Reactive search algorithms modify algorithm parameters and/or strategies during the search through a feedback mechanism which uses information about the search history (Battiti & Brunato, 2007, chap. 21). These algorithms can adapt to the local characteristics of the search space.

One of the best known reactive search algorithms is Reactive Tabu Search (Battiti & Tecchiolli, 1994), which is a modification of tabu search. The tabu search algorithm searches the solution space by applying moves to a solution in order to produce a new one while forbidding the reversal of these moves for a certain number of iterations. Reactive Tabu Search uses a feedback mechanism to determine the number of iterations during which reversing a move is forbidden.

The Guided Local Search approach (Voudouris & Tsang, 1999) guides the search by adjusting the penalty parameters, which are added to the objective function value. Some genetic algorithms (Eiben, Hinterding, & Michalewicz, 1999) use feedback to control the mutation parameters. Another reactive search strategy involves adapting the greediness of the search process according to feedback coming from the search (Hoos, 2002). The improvement, however, significantly depends upon the type of the problem being solved.

In engineering, reaction mechanisms are based on *control theory* (Franklin, Powell, & Emami-Naemini, 2002), which provides a systematic approach to the design and analysis of the adaptation process. In this research, we contribute to the area of reactive search with two reaction strategies. The first strategy uses a control-theoretic approach, in which a controller is used to keep outputs of a dynamical target system at a desired level. In this study, we treat the tabu search algorithm as a target system to be controlled, and use control theory to design a feedback-based reaction mechanism to improve the performance of the search by

☆ This manuscript was processed by Area Editor Ibrahim H. Osman.

* Corresponding author. Tel.: +90 3122924278; fax: +90 3122924091.
E-mail addresses: nfunver@etu.edu.tr (N. Fescioglu-Unver), mkokar@ece.neu.edu (M.M. Kokar).

0360-8352/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2010.11.014

controlling its intensification. The second strategy is a diversification mechanism, which adapts algorithm parameters according to feedback coming from the search history. These two strategies work together to form a new tabu search algorithm – Self Controlling Tabu Search (SC-Tabu). The goal of this paper is not to develop the most efficient search algorithm for a specific problem domain. Rather, the goal is to show that control theory principles can be used to design reaction mechanisms for heuristic algorithms.

We implemented the SC-Tabu algorithm for the NP-hard Quadratic Assignment Problem (QAP) (Sahni & Gonzalez, 1976). Many industrial problems, such as airport gate assignments, factory/office layout design, image processing, network design and printed circuit board design can be modeled as QAP; therefore, the QAP is a good testing domain. In our experiments, we used benchmark problems from the Quadratic Assignment Problem Library (QAPLIB) (Burkard, Çela, Karisch, & Rendl, 2008).

In Section 2, we present short descriptions of tabu search algorithm, intensification and diversification notions, the self-controlling software concept, and the Quadratic Assignment Problem. Section 3 introduces the Self Controlling Tabu Search algorithm. The experimental results with the problem domain and comparisons are presented in Section 4. Finally, Section 5 discusses overall results and future research directions.

2. Background

In this section, we present information about the building blocks of the developed algorithm. We also discuss the Quadratic Assignment Problem, which is the application and test domain of our algorithm.

2.1. Tabu search

Tabu search (TS) was introduced by Fred Glover (Glover, 1989, 1990a, 1990b) as an iterative heuristic-based algorithm for solving combinatorial optimization problems. TS searches the solution space in *iterations*. In each iteration, TS generates a solution *neighborhood* via *moves*. Each move is a transformation that leads from one solution to another and TS keeps the best solution.

The distinctive property of TS is its use of search history (*memory*) to prevent search from cycling back to previously visited solutions. The search history is kept in the memory, called *tabu list*. Tabu list keeps either some of the moves or just their attributes. Reversing these moves is forbidden for a given number of iterations, called the *tabu tenure*. Any moves that would undo these transformations are listed as *tabu*. However, if a tabu move meets the *aspiration condition*, also called the *aspiration criterion* (e.g. it hitherto provides the best solution), then the move is allowed.

2.2. Intensification and diversification

Search algorithms often incorporate two strategies: greedy and exploratory. In a minimization problem, the greedy strategy drives the search toward the steepest decrease in the value of the objective function. This, however, may result in finding local optima, instead of global. The exploratory strategy forces the search to different regions in the search space so as not to miss the global optimum.

Similar ideas are captured by the notions of *intensification* and *diversification*. Intensification refers to focusing the search within an attractive solution region, whereas diversification refers to driving the search to explore unvisited regions (Glover & Laguna, 1998).

When such opposing strategies like intensification and diversification are implemented, the balance between them is determined by certain parameters of the algorithm. Most combinatorial optimization problems have different solution distribution characteristics within the search space. This creates unique requirements for intensification and diversification which must be customized to each situation. Using the same intensification and diversification parameter settings for all types of problems would disregard a specific problem's needs. Moreover, these requirements not only differ from problem to problem, but also often change in the course of solving a single problem. In some problems, good solutions are distributed more or less uniformly within the search space, while in others good solutions are found in clusters (localities). Even if the parameters are tuned for a specific problem instance, using the same settings throughout a search leads to intensifying or diversifying either more or less than needed.

2.3. Self-controlling software

Self-adaptive software is software that modifies itself in runtime to achieve better performance. The idea of self-modifying software has been explored for some time now. For instance, the discipline of machine learning develops software that modifies itself in response to (typically) external feedback. Kokar, Eracar and Baclawski (Eracar & Kokar, 1997; Kokar & Eracar, 2000; Kokar, Baclawski, & Eracar, 1999) introduced the idea of *self-controlling software* in which software can be considered as a *target system* with its own dynamics and can be controlled with feedback based on a Quality of Service function. A similar idea was later independently introduced by Herring (Herring, 2002). Herring's concept was called the Viable Software Approach (VSA). It was modeled after Stafford Beer's Viable System Model (VSM) (Beer, 1981). In software engineering, similar ideas are known as *active software*, *self-adapting software* (Laddaga, 1999; Robertson & Laddaga, 2004), *software cybernetics* (Kai-Yuan, Cangussu, DeCarlo, & Mathur, 2003) and *autonomic computing* (Kephard & Chess, 2003). VSM was also used as part of the design methodology for autonomic systems (Taleb-Bendiab, Bustard, Sterritt, Laws, & Keenan, 2005).

In this paper, we follow the approach described by Kokar et al. (1999), which uses control theory principles to control the software parameters. Control theory is used in engineering to regulate mechanical, electrical, chemical, and computing systems. The essential elements of a feedback control system are as follows:

- *Target system*, also referred to as *Plant*: The system to be controlled.
- *Controlled output*: A characteristic (a variable) of the target system that is controlled.
- *Control input*: A variable that affects the controlled output.
- *Reference input*: The desired value of the measured output.
- *Disturbance*: Variables that affect the measured output but are not controlled.
- *Control error*: The difference between the value of the reference input and the measured output.
- *Controller*: An equation (referred to as the *control law*) that determines the control input value needed in order to achieve the reference input.

The self-controlling software approach identifies the software as a target system whose efficiency can be improved by dynamic adjustments provided by a feedback-based controller (Kokar et al., 1999).

2.4. Problem domain: Quadratic Assignment Problem

The Quadratic Assignment Problem can be described as a logistics problem where n units are assigned to n different locations. For