Descending Deviation Optimization Techniques For Scheduling Problems

Kevin McCarty
University of Idaho at Idaho Falls
1776 Science Center Drive
Idaho Falls, ID 83415, USA
mcca6934@vandals.uidaho.edu

Milos Manic
University of Idaho at Idaho Falls
1776 Science Center Drive
Idaho Falls, ID 83415, USA
misko@ieee.org

***Abstract*— In factory automation, production line scheduling entails a number of competing issues. Finding optimal configurations often requires use of local search techniques. Local search looks for a goal state employing heuristics and random local "probes" in order to move from state to state. All local search techniques, however, suffer from problems with local maxima, i.e. have the potential of getting "stuck" in a suboptimal state. While careful introduction of randomizations is certainly a recognized technique, it can also lead the algorithm even more astray. This paper describes a heuristic technique called Descending Deviation Optimizations (DDO) in which a gradually lowering-- randomization ceiling allows a local search technique to "bounce" randomly without going too far astray. An example applying the DDO to a local search technique and achieving significant improvement is shown.**

I. INTRODUCTION

One of the very important aspects of factory automation is the efficient use of both time and space [6], [14]. Doing so requires the precise coordination of people, material and equipment in limited space boundaries in order to maximize throughput and minimize latency [11]. However, having one optimal set layout is generally impractical as priorities often change during the course of a production cycle or significant events [17]. For example, the breakdown or introduction of new equipment can significantly affect the production schedule.

Unfortunately, combinations of variables and constraints can quickly result in the factorial growth of the possible permutations to search beyond the practical ability of modern computer systems to thoroughly assess. An exhaustive search through a space of potential configurations becomes impractical. Problems like that of the production scheduling problem mentioned above belong to a generic class of problems called Constraint Satisfaction Problems (CSPs). CSPs often encompass a potential set of states for which the entire state space is beyond a system's ability to search comprehensively. CSPs belong to a class of combinatorial problems called NP for "Non-Deterministic Polynomial" for which a given solution can be found by a polynomial-time algorithm [1], [2].

Local Search Algorithms (LSAs) have proven very useful for finding solutions to CSPs [12]. LSA compensate for a lack of universal awareness by starting at some beginning state then exploring neighboring states, testing for goal states along the way [3]. This allows for a smaller requirement of resources as only neighboring states need to be stored or searched. If there are multiple goal states in the overall state space, then there is a significant probability that an LSA will discover one quickly. This makes LSAs a preferred method for solving CSPs such as the factory automation production scheduling problem [11]. However, there are many different LSA techniques and all have various issues particularly dealing with locally optimal but globally sub-optimal states called local maxima. Descending Deviation Optimizations addresses some of these issues by allowing LSAs to move away from local maxima but in a controlled fashion so as to have a higher likelihood of finding global maxima.

This paper is organized as follows: Section II presents a problem statement and a brief look at various local search techniques under consideration. Section III presents the Descending Deviation Optimizations steps. Section IV presents the application of Descending Deviation Optimizations to Simulated Annealing and Stochastic Hill Climbing to solve the scheduling problem along with other test results for many of the traditional techniques. Section V presents conclusions and future work.

II. BACKGROUND AND PROBLEM STATEMENT

Factory automation scheduling must often take into account a myriad of competing elements ranging from floor space to personnel and equipment costs, production times to shipping and delivery priorities. Finding the best configuration means creating a schedule where conflicts and idle time are reduced to the greatest extent possible [13], [14], [15].

In a typical factory a number of different products are produced. Each product consists of components. Each component is made from raw materials or other components delivered to the factory or produced within it. Raw materials must be shipped in and moved across the factory floor from a receiving point to a production station navigating a maze through which other raw materials, components and products must also pass.

For testing purposes, consider a factory that produces 8 products. From raw material to shipped product there are 8 stations to pass through: receiving (R), component assembly (CA), quality inspection (QI), final assembly (FA), inventory control (IC) and storage (S), testing (T) and loading and shipping (LS).

The factory's goal is to try to reduce the time material is in the plant to the smallest possible amount. This means moving raw materials in and product out as quickly as possible. But the situation can be very dynamic, with changing schedules and priorities and product lines. In order to work as efficiently as possible the factory must be able to create a layout that will allow materials to move such that there is a limited amount of time spent waiting to move to the next station. Constraints include minimum time spent at each station and minimum time to move between stations. The goal state was one in which all products were at a given station with no product waiting on another.

As there was no way to predict what the initial state would be, a random state generator was used to create 1000 random starting states to see how well the scheduling problem could be resolved to a goal state.

A number of Local Search Algorithms (LSAs) were used to see which could most consistently move from a random starting state to a goal state without being trapped by local maxima.

Heuristics determine where in the local neighborhood LSAs are to search as well as places to avoid. Many LSAs work to explore nearby maxima through a process of moving to successively more optimal states, hoping to encounter a global solution along the way [1], [3], [5]. The problem is that many problems are populated with localized maxima such that flowing nearby gradients can "trap" a LSA by leading it to a position which is not a global solution but in which all of its neighboring positions lead to a less optimal state as shown in Fig 1.

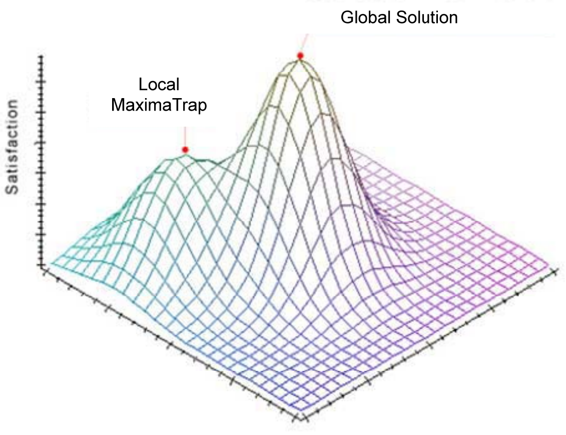

Fig. 1. Comparison of Local/Global Maxima

Some LSAs attempt to escape out of these local maxima through some sort of random "bounce" [7]-[10] which moves an algorithm to a less optimal state but potentially into a location more capable of providing a solution, shown in Fig 2.

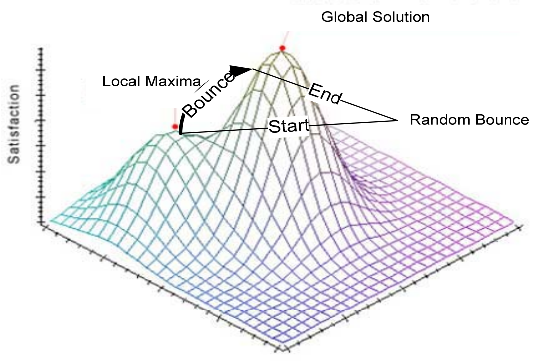

Fig. 2. Bounce out of a Local Maxima Trap

These random "bounces" are often not successful however, leading an algorithm away, rather than towards, a solution, expending time and computing resources in a fruitless search. Descending Deviation Optimizations (DDOs) tries to improve upon an LSAs ability to escape local maxima and find a goal state by restricting it movements somewhat in order to prevent it from moving to far away in any random direction from a potential goal state. This process works well in the scheduling problem presented in this paper because the state space contains a number of goal states spread throughout.

Local search algorithms tried were as follows:

1. Hill Climbing
2. Stochastic Hill Climbing
3. Random Restart Hill Climbing
4. Simulated Annealing
5. Genetic Mutation
6. Minimum Conflicts Search
7. Tabu Search

The results are listed in Table 1.

TABLE I. INITIAL RESULTS OF LSA TESTING

Algorithm	Tries	Success	Failure	% Success
Hill Climbing	1000	141	859	14.1
Stochastic Hill climbing	1000	146	854	14.6
Random Restart Hill Climbing[1]	1000	866	134	86.6
Simulated Annealing [2]	1000	271	729	27.1
Genetic Mutation[3]	1000	229	771	22.9
Min Conflicts[4]	1000	919	81	91.9
Tabu Search[5]	1000	680	320	68.0

1. Random Restart declares failure after 100 iterations and no goal state
2. Simulated Annealing alpha set at .99, number iterations max set to 1000, temp set to 1000
3. Genetic Mutation population of 30 boards, sample of 2
4. Min Conflicts max iterations set to 100
5. Tabu Search max iterations set to 100

All of the techniques had some success in finding goal states, but the most successful required additional memory resources (Minimum Conflicts Search, Tabu Search) or a "lucky" combination of start states (Random Restart Hill Climbing) in order to succeed. With limited resources, it would be more advantageous to implement a different strategy using one of the other techniques and the Descending Deviations Optimization.

III. THE DESCENDING DEVIATION OPTIMIZATIONS TECHNIQUE

Steps in the Descending Deviation Optimization (DDO) Implementation are then as follows:

Step 1. DDO-LSA generates a potential random choice. If the choice leads to a goal state then declare success.

Step 2. DDO-LSA choice is compared to the DDO threshold. If the choice moves the algorithm beyond that threshold, then choice is rejected and algorithm selects another random choice and tests again until a choice if found or all choices are tested.

Step 3. If choice is accepted, the optimal threshold reduced by a predetermined amount and the algorithm moves to Step 1.

As an example of the DDO technique consider a common local search technique: Simulated Annealing.

Simulated Annealing (SA), named after a process in metallurgy whereby metals are successively heated and cooled, implements a succession of random "bounces" that slowly diminish over time [9], [10]. SA's pseudo-random selection method measures a random pick against a slowly descending de-optimization threshold. The algorithm allows a large range (nearly random) set of choices early on, getting progressively more restrictive in favor of better choices with each iteration. Since the range of options is greater in the beginning, it will have a tendency to explore more maxima and is correspondingly more likely to find one that is a global solution. SA is able to explore a relatively wide range of possibilities when compared to other algorithms and does a comparatively good job of finding global maxima compared with other local search techniques. However this can be computationally expensive. In addition, the algorithm can have a tendency to be lead hopelessly astray by a succession of less than optimal choices as demonstrated in Fig 3.

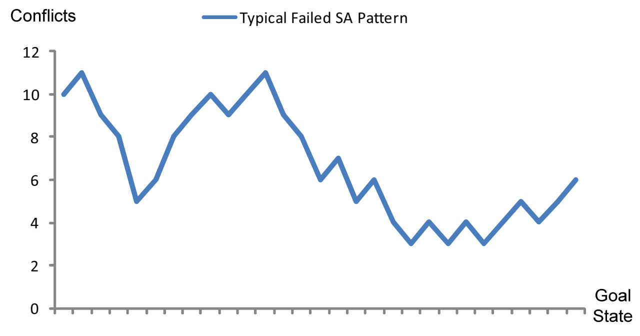

Fig. 3. Pattern in which SA fails to find a solution

The DDO approach to SA takes the original SA implementation and adds the following optimizations:

1. An artificial, decreasing ceiling is imposed on the allowable number of conflicts. The DDO threshold is a function of the square root of the temperature variable. The square root function is chosen in order to provide a smooth, regular ceiling descent that eventually converges to the original SA threshold as they both approach zero. The ceiling prevents the solution from going from a lower state to a much higher state late in the process via a series of small, negative changes demonstrated in Fig 4. With each iteration the DDO threshold forces the SA to explore a smaller and smaller range of randomizations, hopefully to move it more quickly to the goal state.
2. Some versions of SA pick a value and may or may not use it depending upon whether or not it exceeds some "fitness" value. In this case, all the local potential moves are tested. Any move which would cause a no-op to occur is thrown out of the sample of choices so that each iteration produces only those values that meet the fitness criteria.
3. During the screening process, if a tile is found that reaches the goal state, use that tile automatically so the process ends in success.

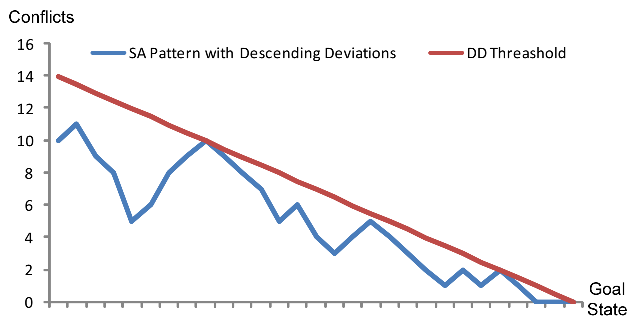

Fig. 4. Simulated Annealing with Descending Deviations

IV. Test Examples

In order see how effective DDOs are, the best and worst LSAs were chosen for implementation; Simulated Annealing (SA) as the best performing and Stochastic Hill Climbing (SHC) as the worst. Traditional Hill Climbing does not utilize a random component so was excluded.

Stochastic Hill Climbing (SHC) is a variant of the traditional Hill Climbing in which not the steepest ascent is picked but any ascent is eligible, dictated by a probability assigned to each option [3]. The probability is dependent to some degree upon the steepness of the ascent.

DDO-SHC works exactly like the traditional SHC until it gets "stuck", at which point it "bounces" the solution to a nearby, less optimal state and again applies the original strategy. The "bounces" are gradually lessened in height or until they disappear at which time if a global solution is not reached, the strategy fails.

The DDO-SHC and DDO-SA were added to the suite of LSAs and tested against the scheduling problem. The results of the modified LSAS are listed in Table II.

Table II. Results of Modified Local Search Algorithm Testing

Algorithm	Tries	Success	Failure	% Success
DDO-Stochastic Hill Climbing[1]	1000	253	747	25.3
DD-Simulated Annealing[2]	1000	993	7	99.3

1. DD-Hill rescued 121 failures, threshold set at 5
2. Simulated Annealing/DD-Simulated Annealing alpha set at .99, number iterations max set to 1000, temp set to 1000

In both cases DDO modifications to the original LSAs resulted in significant improvements in the LSAs ability to avoid local maxima and find a global solution. The DDO-SHC success rate nearly doubled (14.6% to 25.3%) while the DDO-SA achieved an almost fourfold (27.1% to 99.3%) rate increase to the point it was nearly perfect and better than any of the traditional LSAs tried.

There were 2 additional benefits as well for the DDO-SA algorithm. Despite the additional overhead imposed by the DDO, the increased success rate resulted in fewer iterations overall for the given 1000-test cycle. In addition, the algorithm also displayed a lesser tendency to "wander around" or be lead astray by a series of bad choices. This resulted in both more successes and 20% fewer iterations overall than traditional SA to reach the goal state. This lead to a net time reduction of over 35% to complete 1000 iterations, also resulting in a large net decrease in computational resources required.

V. Conclusion and Future Work

The results demonstrate the significant advantages of Descending Deviation Optimizations to Local Search Algorithms in production line scheduling problems of factory automation. Compared with the traditional approaches, DDO-Optimized versions were many times more successful overall in finding a solution. In the case of DDO-SA, the descending ceiling prevented many fruitless searches due to a lengthy series of bad choices resulting in a 99% decrease in the number of unsuccessful attempts compared to traditional SA. This resulted in a significant net time reduction and large net decrease in computational resources required. DDO-SA also outperformed all other local search algorithms tested, including those heavily dependent upon memory resources.

Of particular note, the "lucky" LSA, Random Restart Hill Climbing, given enough restart chances, would be more successful at reaching the goal state than SA-DDO. However, the number of restarts, and corresponding resources and time needed, will be significant as even 100 restarts still resulted in a failure rate nearly 20 times greater than SA-DDO.

Future work is to apply the DDO method to the other LSAs to see which sorts of improvements are possible elsewhere. It appears possible that any of the traditional LSA techniques, such as Genetic Mutation, which utilize some randomization can be improved. Other algorithms for producing a descending ceiling need to be tested for effectiveness. Also to be tested is how DDO-SA will work in place of SA for other classes of NP-problems such as neural network optimizations, adversarial game play, intelligent control and chip layout and whether results are comparable to those achieved in the factory scheduling problem.

References

[1] I. Martinjak, M. Golub; *Comparison of Heuristic Algorithms for the N-Queen Problem*; 29th International Conference on Information Technology Interfaces, June 2007

[2] M. Sipser; *Introduction to the Theory of Computation, 2nd Ed*; Thompson Course Technology 2006, pp 264-269

[3] S. Russell, P. Norvig; *Artificial Intelligence – A Modern Approach, 2nd Ed.*; Prentice Hall 2003, pp 95-114

[4] M. Gao, J. Tian; *Path Planning for Mobile Robot Based on Improved Simulated Annealing Artificial Neural Network*; 3rd International Conference on Natural Computation, 2007

[5] A.H. Mantawy, Y.L Abdel-Magid; *Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem*; IEEE Transactions on Power Systems, 1999

[6] S. Zheng, W. Shu, L. Gao; *Task Scheduling using Parallel Genetic Simulated Annealing Algorithm*; IEEE International Conference on Service Operations and Logistics, and Informatics, 2006

[7] K. Kurbel, B. Schneider, K. Singh; *Solving optimization problems by parallel recombinative simulated annealing on a parallel computer-an application to standard cell placement in VLSI design*; IEEE Transactions on Systems, Man, and Cybernetics, Part B, 1998

[8] V. Pasias, D.A. Karras; R.C. Papademetriou; *Traffic engineering in multi-service networks comparing genetic and simulated annealing optimization techniques*; IEEE International Joint Conference on Neural Networks, July 2004

[9] G. Kliewer, S. Tschoke; *A general parallel simulated annealing library and its application in airline industry*; International Parallel and Distributed Processing Symposium, 2000

[10] P.R.S Mendonca, L.P Caloba; *New simulated annealing algorithms*; IEEE International Symposium on Circuits and Systems, 1997

[11] B. Liu, L. Wang, Y.H. Jin; *An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling*; IEEE Transactions on Systems, Man, and Cybernetics, February 2007

[12] F.G. Guimaraes, F. Campelo, H. Igarashi, D.A. Lowther, J.A. Ramirez; *Optimization of Cost Functions Using Evolutionary Algorithms With Local Learning and Local Search*; IEEE Transactions on Magnetics, 2007

[13] S. Li, Y. Li, Y. Liu; *Effects of Process planning Upon Production Scheduling under concurrent Environment*; The 6th World Congress on Intelligent Control and Automation, 2006

[14] R.B. Chase, F.R. Jacobs, N.J. Aquilano; *Operations Management for Competitive Advantage, 11th Ed*; McGraw-Hill, 2006, pp 22-50

[15] H. Ishibuchi, T. Murata; *Local search procedures in a multi-objective genetic local search algorithm for scheduling problems*; International Conference on Systems, Man, and Cybernetics, 1999

[16] H. Ishibuchi, T. Yoshida, T. Murata; *Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling*; IEEE Transactions on Evolutionary Computation, 2003

[17] M.E. Garcia, S. Valero, E. Argente, A. Giret, V. Julian; *A FAST Method to Achieve Flexible Production Programming System*; IEEE Transactions on Systems, Man, and Cybernetics, 2008