Design Engineering

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

Cost Optimization at Early Stages of Design Using Deep Reinforcement Learning (DRL)

Author Name:
Naresh Babu Bynagari
Affiliation:
Director of Sales, Career Soft Solutions Inc, 145 Talmadge rd Edison NJ 08817, Middlesex, USA

Abstract

There has unarguably been an increase in how complex modern systems are when it comes to Chips (SoCs). This, coupled with the rising demand for a time-to-market provision lower than usual, automation assumes an ultimately essential component in designing hardware. As a matter of particular relevance, this comes in handy for tasks that are time-consuming or overly complex in nature. By optimizing the cost of design for any hardware component, automation becomes an effective reality. In fact, design cost can be reliant on a number of objectives, in semblance to the trade-off between the hardware and the software. Because this task can often be multiplexed, the designer in charge will have little to no means of delivering timely and efficient optimization for the even larger and more compound models. This paper initially demonstrates that the DRL is an ideal solution for the problem encountered in this process. Thereafter, using a Pointer Network, which is a system of neural elements painstakingly tailored to play a role in the application of combinatorial complexities, we measure a trio of DRL algorithms against a specified challenge. The outcomes realized in the many cases showcased the developments that have occurred by the said DRL algorithms in comparison to traditional models for optimization. Furthermore, through the use of reward re-dispensation suggested in the recently published RUDDER technique, the paper garners substantial betterments on the part of complex designs. Herein, the average optimization obtained is 15.18 percent area-wise. On the application size, the average is 8.25 percent, while being 8.12 percent on the executive time. This happens with industrial hardware-cum-software interface design data sets.

Keywords: Cost Optimization, Deep Learning, DRL, Artificial Intelligence

INTRODUCTION

The rapid development of Artificial Intelligence (AI) has been brought on by the significant progress of computer networks as well as hardware. However, as Moore's Law and Dennard scaling to an end, the glove is now experiencing a significant shift towards specifically tailored hardware in order to meet Artificial Intelligence's exponentially increasing demand for computer systems. Be that as it may, the chip designs of today tend to take years to come up with. That brings us to the speculative task that is the optimization for machine learning models of the next

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

2 to 5 years. When the design of a chip has a substantially shortened cycle, it would be possible for the hardware to adapt more rapidly to the equally rapidly developing field of Artificial Intelligence.

In this research, it is our belief that Artificial Intelligence, by itself, will provide the means through which the cycle of a chip design can be truncated. That puts the hardware and Artificial Intelligence in symbiotic relations with each other, thus turbocharging the advances on the latter. With this paper, we introduce a training-based method for placing chips, which happens to be one of the most complicated and time-consuming processes in designing chips. Placing a netlist macros graph like the SRAMs and standard cells—logic gates like NAND, XOR, and NOR—onto the canvas of a chip in such a manner that the power, performance and area (PPA) will be optimized (Shahookar Mazumder, 1991). That will occur while obeying the ramifications that exist around the density of placement and routing congestion.

It is quite important to improve on how optimizers off the shelf perform, mostly for the time-challenges optimization drawbacks. Case in point, the LMA algorithm (Pollefeys et al., 1996) has become a popular choice for a good number of real-time computer vision challenges (. That includes tracking objects using video enablers—where only a small amount of time can be allocated to the optimizer on every emerging frame of video. In fact, time-limited optimization has become an increasingly critical problem when it comes to applications like machine perception, operations research and robotics.

For these problems, the objective is to achieve the maximum solution in a specific time frame. Considering the unique attributes of time-constrained issues, there is a likelihood that the heuristic-dependent controllers employed in the off-shelf optimizers may not deliver particularly impressive results. In addition, touchstone nonlinear optimization methods such as LMA are not in the capacity to address challenges like the stop time for an unrewarding native search or the right time to visit a part of the parameter that was recently visited.

What is Reinforcement Learning (RL)? It is a machine learning method which involves training optimal controllers using case studies. This learning method is, apparently, an ideal candidate when it comes to the improvement of the heuristic-dependent controllers put into use in the most widespread and most used algorithms for optimization. The upper hand RL models have over the other optimal control methods is not requiring initial knowledge of the subsisting dynamism in the system. Plus, the designer is at liberty to select the reward metrics that most complement the desiderata for the delivery of the controller. For instance, when optimizing under time constraints, an ideal reward can be achieving the lowest possible loss within a set period of time.

REVIEW OF RELATED LITERATURE

The concept involving the use of RL in optimization is not a novel one (Ganapathy, 2021a; Zhang, 1998; Gambardella and Dorigo, 1995; Boyan and Moore, 1998; Moll et al., 2000).

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

Nonetheless, the previous approaches have shed more light on using RL models in the development of problem-focused optimizers for NP-holistic issues. In this research, we have a focus on leveraging the RL techniques to modify the controllers that are implicit in the most prevalent and most used algorithms for optimization. Our particular goal is to make sure the considered algorithms become more efficient to carry out optimization on problems that are on a time budget.

As will soon be demonstrated, a basic RL method can culminate in substantial developments in the performance of these well-known packages for optimization. There is also existing research on the LMA approaches' empirical evaluations as against other nonlinear optimization models in the computer vision ecosystem (Paruchuri, 2021; Cristinacce and Cootes, 2006). In a study (Lourakis and Argyros, 2005), the LMA approach is compared with Powell's dog-leg technique based on the bundle adjustment issue.

Global placement is no new challenge when it comes to chip design. It requires optimization with a multiplicity of objectives. Since the turn of the 1960s, a multitude of models have been introduced. So far, they all have fallen into a trio of classes, the first of which is the partitioning-based application. The second and third categories are the stochastic or hill climbing method and the analytic solvers, respectively. From the 1960s, it has been the partitioning-based approach to the general placement issues (Ganapathy, (2021) that has been practiced by academic and industrial laboratories. That proposes (Khan et al., 2021; Kernighan, 1985; Fiduccia & Mattheyses, 1982) the resistive network-dependent applications (Bynagari, 2018). These approaches are defined by a divide-and-conquer method wherein the netlist as well as the canvas on the chip is partitioned in a recursive fashion until problems that are small enough emerge.

At this point, the subsisting netlists are deposited in the sub environments with the aid of optimal problem solvers. Such methods are incredibly fast to implement, while being able to habitually scale to more sizable netlists due to their hierarchical natures (Vadlamudi, 2021). Be that as it may, when each of the sub-issues are optimized, the partition-based applications give up the quality of the general solution—particularly when it comes to routing congestion. Additionally, a poor-performing early partition can culminate in end placement (Kirkpatrick et al., 1983) that cannot be salvaged.

TRAINING MODERATION POLICIES FOR OPTIMIZATION ALGORITHMS

The tractability of learning for a controller is of utmost importance. To ensure this, it remains critical to compress the whole history of locations that were visited when optimization was ongoing to a handful of enough measurements referred to as the state. For this paper's purpose, we opted to maintain a restriction in the state of space to a few numbers that accurately symbolize the most recent time limitations and the current changes evident in the optimization process. The space wherein the action takes place is restricted through observing that the ongoing approaches to nonlinear optimization make for salient propositions for the next location

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

to reach (Ganapathy, 2020a). By this means, the action space in our research is able to encode every fixed set data of optimization sub processes for the following iteration. That goes along with the actions that govern all the heuristic parameters in each sub-routine optimization.

Case in point, schedules meant to update η in terms of gradient descent as well as the heuristics factored into the modification of the value of λ in the LMA environment (Vadlamudi, 2020). Also, to better define the optimality of a controller, our research defines an outcome function that is indicative of the demand for the solution created while optimizing. Contextually, optimizing with quasi-rigid time restrictions and an adequate rewarding system creates a balance in the reduction of loss on the part of the objective controls. This is done with some steps that are needed to realize the said reduction. In the optimization that involved a fixed budget, the more natural choice could be the general decrease in the loss function inside the allotted cut of function-facing evaluations. In cases of particular application, with semblance to the spirit work of Boyan (Boyan, 1998), the reward system can be amended to cater to features of transitional solves most likely to be indicative of the desiredness of the ongoing location.

With the state space, reward function and action space for a particular optimization task, reinforcement training approaches make for an ideal set of methods for training the controller of an optimization system. There may be countless algorithms for reinforcement learning that are just about suitable for the formulation of our problem, we favor the Least-Squares Policy Iteration (Bynagari, 2017). LSPI is quite the attractive deep learning algorithm because it is capable of handling continual state spaces. What's more, the algorithm is effective when it comes to the amount of interaction it can have with the network in order to learn suitable controllers. It's also the ideal select given that it has no dependency on the subsisting model of the procedure's metamorphosis. LSPI can also learn models that can be modified based on interpretations.

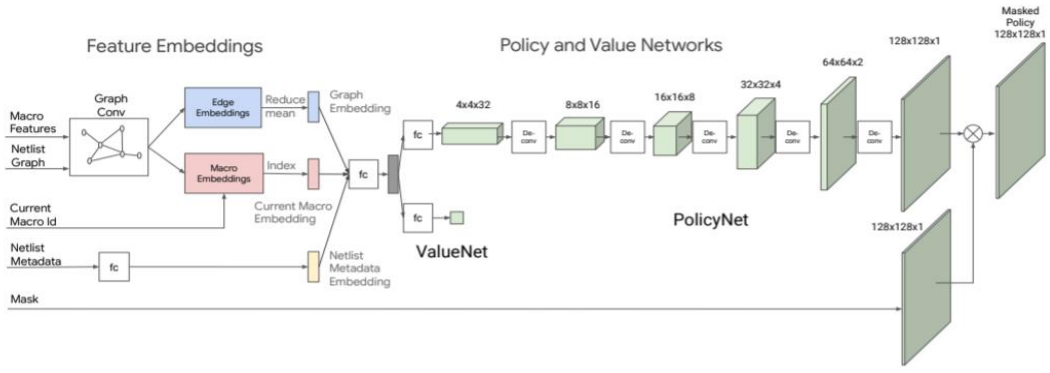

Figure 1: LSPI procedure with two steps of feature embedding and value functions.

The LSPI procedure is not only quite iterative but also can repeatedly apply these two steps until it reaches a point of convergence. The first step involves the approximation of the action-value function in terms of a linear blend of non-modifiable set of basis controls. The second step is the

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

greedy improvement of the ongoing policy over the estimated value function (Figure 1). Bases represent the functions of the state while the action can assume nonlinear forms. This application comes in handy for the amount of interactions needed with the system of dynamics. Plus, it can reuse a set of examples for the evaluation of countless policies. That is a critical feature, given the need to ascertain the difference between the LSPI and the preceding applications such as the LSTD. The LPSI procedure's output is a weight vector that interprets the action-cum-value function of the optimized policy as a linear amalgamation of the rudimentary vectors.

Our approach for training the controller of an optimization system comprises two stages. In the first stage, we collect samples via interactions between a randomly selected optimization controller and a problem that needs to be optimized in a series of optimization sessions with non-adjustable durations. The samples come in tuples form (s, a, r, s0) where s0 stands for the state that is realized during action execution, beginning from the states until when the reward function finally kicks in. In the second stage for learning the optimization controller, our algorithm puts the LSPI into use to train an action-value system. It also implicitly trains an optimal policy which is often provided by the greedy maximization of the said function over the actions that occur for a particular state in the optimization process.

OPTIMIZING NONLINEAR LEAST-SQUARES FUNCTIONS USING A FIXED BUDGET

This paper showcases the capacity of our deep learning method to not only achieve maximum delivery to nonlinear optimization approaches that are off the shelf, but also give insights into the individual policies as well as action-value controls. The classical nonlinear challenges and the superficial recognition assignment were both created with regards to optimization given a rigid budget of evaluation-facing functions. The requirement proposes an inherent reward function wherein L represents a loss function that we attempt to bring to a minimum. Here too, B represents the budget involved in function evaluations, while I stands for indication function. Then, Xopt is the location where there is the least amount of loss visited in the ongoing optimization session.

For controllers, the reward function serves as an encouragement. It is for the controllers that realize huge decrements in the loss inside the station budget of the function evaluations (Ganapathy, 2020). Every optimization challenge assumes the making of a minimizer for the total squares of nonlinear controls. Thus, they are ideally suited to the Levenberg-Marquardt method of optimization. The space of action taken into consideration in our tests comprise modifications to the damping element applied in the LMA. It involves the decision of whether the final descent step should be discarded with a pair of actions that the LMA cannot avail. Included actions are moving onto a fresh random location in the objective function's domain. It also included going back to the best location discovered in the process and delivering a single descent step with the LMA approach while employing the ongoing damping factor.

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

The amount of action performances that are availed in every step is 8. Only various amalgamations of modifications to λ and revisiting actions get 6. The space of state used in enabling the action decision comes with a window history of specific length that encodes whether or not a given previous step decreased or increased the remaining inaccuracy from the initial session. The window, in context, is adjusted to size 2 in the majority of our procedures. Nevertheless, our experiments considered evaluating the interpersonal development that occurs when the window size is set to 1 against 2. In our state space, we also included the number of function-focused assessments remaining in the set budget and a challenge-specific state attribute. Both the action space and state are mapped via a garner of rigid basis controls. The LSP algorithm then linearly combines to approximate the best possible action-value performance.

To proceed, we select a basis that isn't complicated and enables seamless interpretation of the weights that have been trained under the LSP algorithm. With this basis, we are able to independently address each action, thus building up a tuple of basis controls for every action in the space. The classical optimization challenge and outward sign categorization challenge both require the tuple of basis. It involves a two-step history window regarding if the loss witnessed an increase or saw a decrease in the past two modifications alongside the amount of phases left currently in the cost analysis. As for the task concerning recognizing facial expressions, the tuple adds a basis.

To verify the method, we apply the approach to a dataset of standard nonlinear optimization challenges (Bynagari, 2016). In this problem data set, there is a well-known optimization challenge that cuts across a long stretch of nonlinear performances. Cases in point, the Rosenbrock function, the Helical Valley and the Powell Singular function. When it is limited to a 20-function-evaluation budget, our approach can learn a policy. That results in a 7 percent increase in gain compared to the LMA's possible results. This performance is quantified as the summed decrease in losses from the point of origin.

THE CLASSIFICATION OF FACIAL EXPRESSIONS

Box-filter characteristics that were successful at detecting faces (Bynagari, 2015) also proved potential to detect facial expression when boosting methods are put into the combination. The reaction from the box-filter to an image patch is ascertained through weighting the total pixel brightness in several boxes. This is done via a coefficient which is defined by the said kernel that is the bix-filter. In the research, we regard the challenge of attribute selection as a procedure for optimization over an endless parameter space. The space defines an unending group of box-filters that comprise several as those that have been proposed (14) as special scenarios. Every characteristic can be described in terms of a vector with six depicted dimensions.

Then, we trained a detector to determine the absence or presence of a smile. We do this with the amount of pixel intensities that occur on the image patch comprising a new face. This is accomplished by employing the periodical retrogression process known as L.2-boost (Bynagari,

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

2014). With L.2-boost, we were able to create fervent categorizers by modifying the leftovers of the ongoing method above a group of weak trainees, which is, in this case, the features we have parameterized.

L.2-boost chooses a box-filter during each iteration, mostly decreasing the disparity between the current forecasts of the approach and the accurate image labels. Once a feature good enough is discovered, it is added to the incumbent assembly. The L.2-boost trains a linear model to determine the label of the image patches given that every weak learner maintains linear filters on the value of the pixels. The L.2-boost combines the weak learners in a straight manner, after which the LSPI's basic space is augmented for this assignment by adding a basis which specifies the amount of attributes pre-selected by the L.2-boost process.

Afterwards, we put our preferred algorithm on the smile detecting assignment, giving it a subset of 1,000 images from the GENKI—a collection of 60,000 faces from the internet. Figure 2 shows a sample optimization budget for the performance on smile detection. Alongside the data concerning the locality of the faces and their physical attributes, human labelers have considered each image to contain or not contain smiles. For these tests, our aim is forecasting a person's smile with the help of the L.2-boost process we mentioned before.

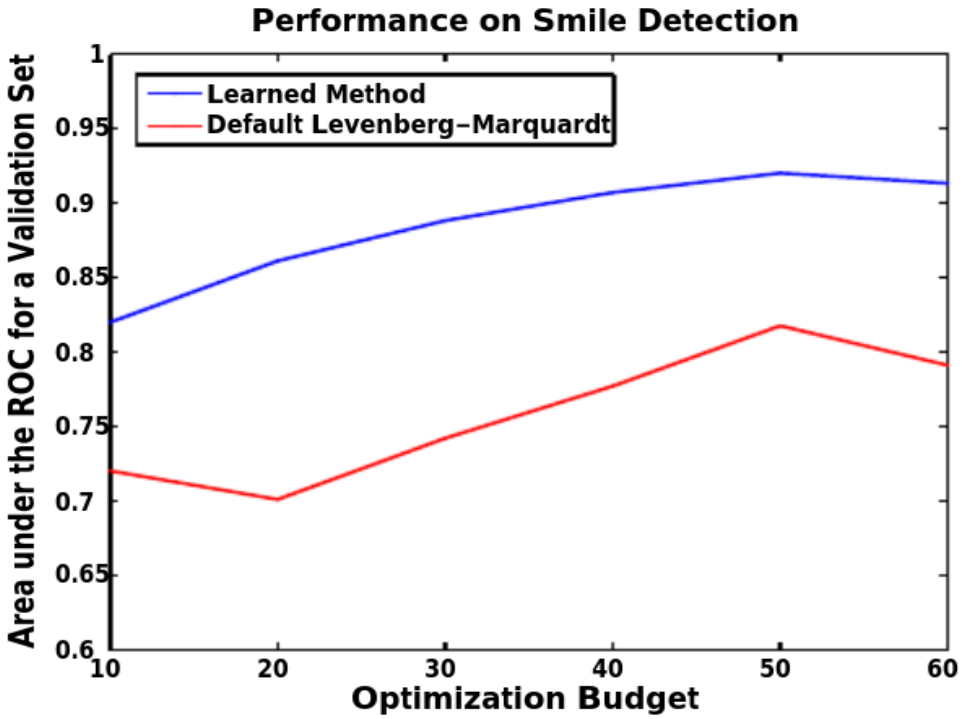

Figure 2: Optimization budget on the performance of smile detection

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

When every trial box filter is chosen with the L.2-boosting. Inside every round-off feature choosing, a sum of 20 attribute evolutions are availed in a single round. So, we implemented the default model of the LMA as a way to compare. After the samples have been garnered from 100 chapters of the GENKI set of data, the LSPI method can train a policy that will realize 3.86 in greater fold decrease compared to the LMA method. Because LMA does not possess access to the capacity to mobilize a new indefinite aspect to a fresh state space, a fairer comparison is our approach with no access to the said action. In our experiment, the method employed is yet capable of reaching 83 percent in more significant reduction in general loss as opposed to the LMA.

OBSERVATION AND CONCLUSION

This experiment has proposed a newer method to the problem associated with the training optimization process when there is a fixed budget. Our study has been able to demonstrate that the method is able to realize a more substantial delivery rate compared to the conventional approach that comprises nonlinear squares. Our provisions also include an analysis of the cycles learned by our approach and the manner in which they make sense when it comes to carrying out problem optimization on a rigid spend plan. In addition, this paper has proposed extensions to the characteristics used in a study (Bynagari, 2014) that are inherently relevant.

Perhaps, in the future, we will embark on an encompassing exploration of the framework we have outlined with this paper. In the current work, the specific application accorded to the said framework, while impeccably efficient, can also be improved on. Case in point, through the incorporation of domain-specific attributes into the space, policies of richer nature would be learned. We as well look forward to applying our model to other issues in the machine perception ecosystem. A subsequent endeavor along these lines will look to test the viability of the technique we proposed for finding the locations of feature points simultaneously exhibiting high chances as regarding looks and relative assembling of facial characteristics. The current limitations of this challenge have made it a specifically ideal target for the applications presented in this paper.

REFERENCES

1. Beardsley, P., Torr, P. and Zisserman, A. (1996). 3d model acquisition from extended image se-quences. Proceedings of ECCV, pp. 683–695.
2. Boyan J. A. and Moore, A. W. (1998). Learning evaluation functions for global optimization and boolean satisfiability, in AAAI/IAAI, pp. 3–10.
3. Bynagari, N. B. (2014). Integrated Reasoning Engine for Code Clone Detection. ABC Journal of Advanced Research, 3(2), 143-152. https://doi.org/10.18034/abcjar.v3i2.575
4. Bynagari, N. B. (2015). Machine Learning and Artificial Intelligence in Online Fake Transaction Alerting. Engineering International, 3(2), 115-126. https://doi.org/10.18034/ei.v3i2.566

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

5. Bynagari, N. B. (2016). Industrial Application of Internet of Things. Asia Pacific Journal of Energy and Environment, 3(2), 75-82. https://doi.org/10.18034/apjee.v3i2.576
6. Bynagari, N. B. (2017). Prediction of Human Population Responses to Toxic Compounds by a Collaborative Competition. Asian Journal of Humanity, Art and Literature, 4(2), 147-156. https://doi.org/10.18034/ajhal.v4i2.577
7. Bynagari, N. B. (2018). On the ChEMBL Platform, a Large-scale Evaluation of Machine Learning Algorithms for Drug Target Prediction. Asian Journal of Applied Science and Engineering, 7, 53–64. Retrieved from https://upright.pub/index.php/ajase/article/view/31
8. Bynagari, N. B., & Fadziso, T. (2018). Theoretical Approaches of Machine Learning to Schizophrenia. Engineering International, 6(2), 155-168. https://doi.org/10.18034/ei.v6i2.568
9. Chen, T., Jiang, Z., Hsu, T., Chen, H., and Chang, Y. (2008). An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(7), 1228–1240. https://doi.org/10.1109/TCAD.2008.923063
10. Cristinacce, D. and Cootes, T. F. (2006). Feature detection and tracking with constrained local mod-els, BMVC, pp. 929–938.
11. Fiduccia, C. M. and Mattheyses, R. M. (1982). A linear-time heuristic for improving network partitions. In 19th Design Automation Conference, pp. 175–181, https://doi.org/10.1109/DAC.1982.1585498
12. Gambardella L. M. and Dorigo, M. (1995). Ant-q: A reinforcement learning approach to the traveling salesman problem, in International Conference on Machine Learning, pp. 252–260.
13. Ganapathy , A. (2021a). Edge Computing: Utilization of the Internet of Things for Time-Sensitive Data Processing. Asian Business Review, 11(2), 59-66. https://doi.org/10.18034/abr.v11i2.547
14. Ganapathy, A. (2020). Virtual Dispersive Network in the Prevention of Third Party Interception: A Way of Dealing with Cyber Threat. ABC Journal of Advanced Research, 9(2), 79-88. https://doi.org/10.18034/abcjar.v9i2.568
15. Ganapathy, A. (2020a). Everything-as-a-Service (XaaS) in the World of Technology and Trade. American Journal of Trade and Policy, 7(3), 91-98. https://doi.org/10.18034/ajtp.v7i3.555
16. Ganapathy, A. (2021). Quantum Computing in High Frequency Trading and Fraud Detection. Engineering International, 9(2), 61-72. https://doi.org/10.18034/ei.v9i2.549
17. Kernighan, D. (1985). A procedure for placement of standard-cell vlsi circuits. In IEEE TCAD.
18. Khan, W., Ahmed, A. A. A., Vadlamudi, S., Paruchuri, H., Ganapathy, A. (2021). Machine Moderators in Content Management System Details: Essentials for IoT Entrepreneurs. Academy of Entrepreneurship Journal, 27(3), 1-11. https://doi.org/10.5281/zenodo.4972587
19. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671–680, https://doi.org/10.1126/science.220.4598.671

ISSN: 0011-9342 | Year 2021
Issue: 6 | Pages: 5238-5247

20. Levenberg, K. (1944). A method for the solution of certain problems in least squares, Applied MathQuarterly.
21. Lourakis, M. I. and Argyros, A. A. (2005). Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?, Proceedings of ICCV.
22. Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters, SIAM Jour-nal of Applied Mathematics.
23. Moll, R., Perkins, T. J., and Barto, A. G. (2000). Machine learning for subproblem selection, in ICML 00: Proceedings of the Seventeenth International Conference on Machine Learning. SanFrancisco, CA, USA, pp. 615–622.
24. Paruchuri, H. (2021). Conceptualization of Machine Learning in Economic Forecasting. Asian Business Review, 11(2), 51-58. https://doi.org/10.18034/abr.v11i2.532
25. Pollefeys, M., Gool, L. V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J. and Koch, R. (2004). Visualmodeling with a hand-held camera. IJCV, 59(3), 207–232.
26. Shahookar, K. and Mazumder, P. (1991). Vlsi cell placement techniques. ACM Comput. Surv23 (2), 143220, https://doi.org/10.1145/103724.103725
27. Vadlamudi, S. (2020). Internet of Things (IoT) in Agriculture: The Idea of Making the Fields Talk. Engineering International, 8(2), 87-100. https://doi.org/10.18034/ei.v8i2.522
28. Vadlamudi, S. (2021). The Internet of Things (IoT) and Social Interaction: Influence of Source Attribution and Human Specialization. Engineering International, 9(1), 17-28. https://doi.org/10.18034/ei.v9i1.526
29. Yogesh Hole et al 2019 J. Phys.: Conf. Ser. 1362 012121
30. Zhang, Y. (1998). Solving large-scale linear programs by interior-point methods under the MATLABenvironment. Optimization Methods and Software, vol. 10, pp. 1–31.

--0--