Brij Disa Centre for Data Science and Artificial Intelligence

IIMA

INDIAN INSTITUTE of MANAGEMENT AHMEDABAD

विद्याविनियोगाद्विकास:

COMMUNIQUE

J U L Y 2 0 2 6

cdsa@iima.ac.in

Brij Disa Centre for Data Science and AI

The Brij Disa Centre for Data Science and Artificial Intelligence housed within the Krishnamurthy Tandon School of AI at Indian Institute of Management Ahmedabad (IIMA) provides a common platform to faculty, scholars, and practitioners for conducting and disseminating cutting-edge research on data analytics and artificial intelligence.

Besides generating action-oriented insights, CDSA is also responsible for dissemination of the knowledge generated for a wider audience both within and outside the realm of the institute. Seminars, workshops, and conferences are regular activities at the Centre, which are conducted to reach out to and engage with multiple stakeholders.

The Centre aims to forge synergistic and collaborative relationships between scholars and practitioners in the field of analytics and data science, besides undertaking case-based research to understand the current industry practice and develop case studies for classroom teaching.

Furthermore, through its collaboration with the industry, CDSA takes up challenging consulting projects of considerable practical importance. These projects are targeted at providing an opportunity for students to participate in projects that aim at outcomes that can further benefit the organization and the business, at large.

A key offering from the Centre is the Annual Research Report, which provides a holistic view of the Data Science and Artificial Intelligence industry, identifies challenges and gaps and offers plausible solutions that can be utilised by practitioners and policy makers.

DRIVING THE GREEN TRANSITION

How Operations Research is Solving the Electric Bus Puzzle

Dr. Pranav Gairola

Centre Research Fellow — Ph.D. Civil Engineering (Transportation) — IIT Delhi

While migrating from diesel to electric public transit is a major environmental milestone, it presents transit authorities with severe operational hurdles. Volatile battery ranges, lengthy charging times, and high-power grid demands mean fleet management cannot remain business-as-usual. This article discusses how Operations Research (OR) and mathematical optimization serve as the mathematical core of this transition—solving interconnected problems in infrastructure layout, vehicle scheduling, and smart charging to maintain and handle city transit efficiently.

Introduction

Electric buses (e-buses) have become an increasingly common sight across major Indian metros like Delhi, Mumbai, and Ahmedabad. Positioned as a critical tool against urban smog and vehicular emissions, public transit electrification is accelerating rapidly. Current projections suggest e-buses will account for a dominant share of new intra-city bus sales in India by 2047 [1].

However, for transit operators, substituting a diesel fleet with an electric one is rarely a straight piece-for-piece swap.

Fleet managers face a variety of structural constraints

- *Steep upfront capital requirements*
- *Limited real-world driving ranges*
- *Inadequate charging infrastructure*
- *Longer charging downtimes*
- *Localized grid strain during peak charging periods*

To overcome these roadblocks, transit agencies are increasingly leveraging Operations Research (OR) as a tool for electric bus fleet management. Through mathematical programming models OR allows operators to evaluate trade-offs systematically. These frameworks optimize route planning, site selection for charging infrastructure, vehicle rosters, and charging schedules simultaneously by minimizing total cost of ownership while meeting strict operational demands.

The E-Bus Dilemma: Why Fleet Electrification is Challenging

The traditional diesel transit network operates on a highly predictable, linear blueprint. A standard diesel bus runs reliably for 16 to 20 hours, covers hundreds of kilometers on a single tank, and can be refuelled in under ten minutes.

16–20 hrs
continuous daily service

100s of km
travel distance per tank

<10 min
quick refuelling turnaround

The traditional diesel transit network

Electric fleets disrupt this baseline due to three distinct operational constraints:

Range Volatility

Unlike fossil fuels, an e-bus battery's real-world range fluctuates heavily based on external factors. High auxiliary loads—such as running air conditioning during an Indian summer—combined with unpredictable traffic congestion can reduce a battery's nominal range by up to 30%.

Operational Downtime

Recharging an e-bus requires anywhere from 30 minutes to several hours depending on the charger type and remaining state-of-charge. This introduces significant idle time into a bus's daily schedule, preventing simple one-to-one replacements of diesel vehicles.

Grid Integration and Infrastructure Bottlenecks

Consolidating high-capacity charging infrastructure within a depot creates massive, localized electricity demand. If a large portion of the fleet plugs in simultaneously at the end of a shift, the resulting demand spike can overwhelm local substations and incur expensive peak-tariff penalties.

***Figure 1**: The classical four-step public transit planning pipeline.*

Rewiring the Transit Planning Pipeline

Historically, transit networks rely on a sequential, four-step planning pipeline to design their operations [Fig. 1]. Each step represents an isolated, computationally intensive mathematical problem:

0 1

Network Design

This phase establishes route paths, stop locations and daily frequencies. Modelled as the Transit Route Network Design Problem (TRNDP), it requires balancing operator costs against passenger convenience metrics like walking distance and transfer frequencies.

0 2

Timetabling

Once routes are defined, planners construct arrival and departure schedules. This is frequently formulated as a Mixed-Integer Linear Programming (MILP) problem to synchronize vehicle arrivals at high-traffic transfer nodes, minimizing passenger wait times [2,3] and preventing real-world issues like bus bunching.[1] [2]

0 3

Vehicle Scheduling

Physical vehicles are assigned to specific trip sequences (blocks). For diesel fleets, this is typically handled via Minimum-Cost Network Flow formulation. Because diesel buses can run continuous, full-day shifts without intermediate refuelling, this has long been considered a settled operational task.[3]

0 4

Crew Scheduling

Finally, drivers are matched to the established vehicle blocks. Modelled using Set Partitioning or Set Covering formulations, this step coordinates driver shifts while strictly respecting labour regulations, maximum driving hours, and mandatory break periods. [4]

[1] That frustrating real-world scenario where you wait ages for a bus, only for two or three to arrive at the exact same time during rush hour traffic.

[2] Note: Think of MILP as a powerful mathematical calculator that sorts through millions of possibilities to make two types of decisions simultaneously: 'yes/no' choices (like whether a bus should charge now or later) and precise numbers (like the exact amount of energy a bus needs).

[3] In simple terms, this model acts like a digital logistics mapper, finding the absolute cheapest way to route a fleet through a web of connecting paths.

[4] In plain terms, these models act like a hyper-advanced jigsaw puzzle solver. The algorithm must piece together thousands of individual driving shifts to ensure every single bus trip is covered without any overlap while respecting all the regulations.

The Electric Vehicle Scheduling Problem (E-VSP)

Electrification breaks this neat, sequential approach [5,6]. Because vehicle availability is directly tied to battery charge status and charging durations, vehicle scheduling can no longer be solved independently of timetabling or crew assignments. Drivers can rarely stay with a single vehicle for an entire shift; instead, they must rotate based on vehicle charging windows.

Consequently, modern OR research is shifting toward integrated, simultaneous optimization frameworks. These systems merge vehicle schedules, charging logic, and driver assignments into a single, unified mathematical model to guarantee operational feasibility [Fig. 2].

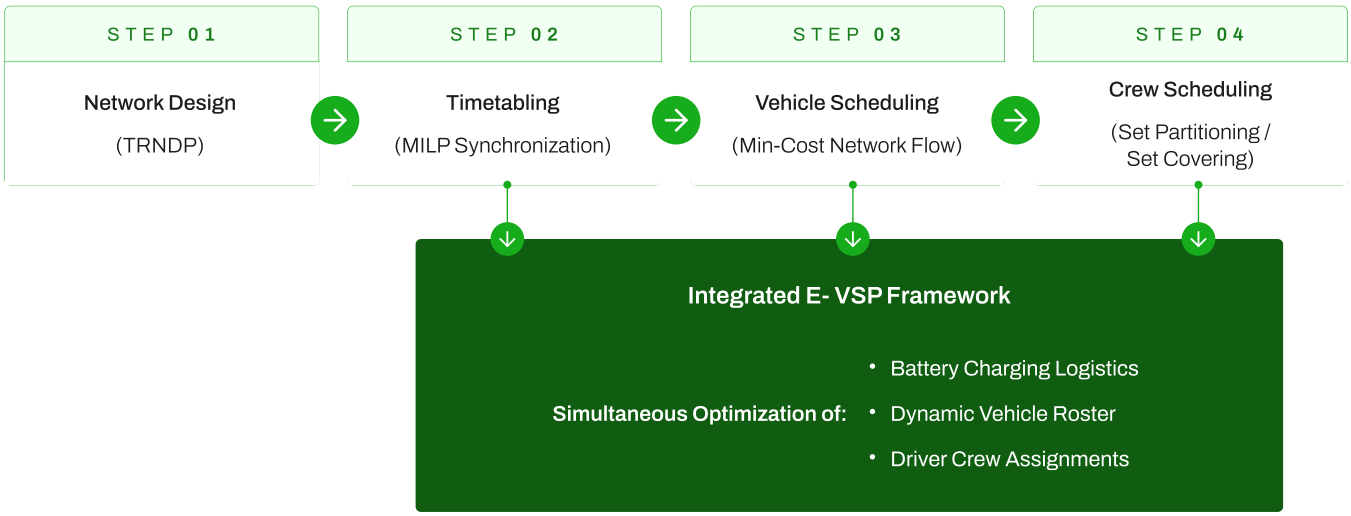


***Figure 2**: The breakdown of the traditional sequential transit pipeline into an integrated, simultaneous E-VSP optimization framework.*

Smart Charging: Balancing the Grid

Fleet energy costs are highly sensitive to dynamic electricity pricing, such as Time-of-Use (ToU) tariffs, which penalize high power consumption during peak hours. Uncoordinated, simultaneous charging of multiple high-capacity batteries creates severe load spikes that can trigger significant financial penalties and compromise depot electrical infrastructure.

OR models address this via Charge Scheduling Optimization. Instead of letting vehicles draw power immediately upon arrival, the optimization framework staggers power delivery based on roster requirements. A bus scheduled for an early 5:00 AM departure is prioritized for fast charging, while a vehicle rostered for 9:00 AM is allocated a slow, overnight trickle-charge when utility rates are lowest. This load-shifting mechanism flattens the depot's power demand curve, avoiding surcharge thresholds and reducing infrastructure strain [Fig. 3].

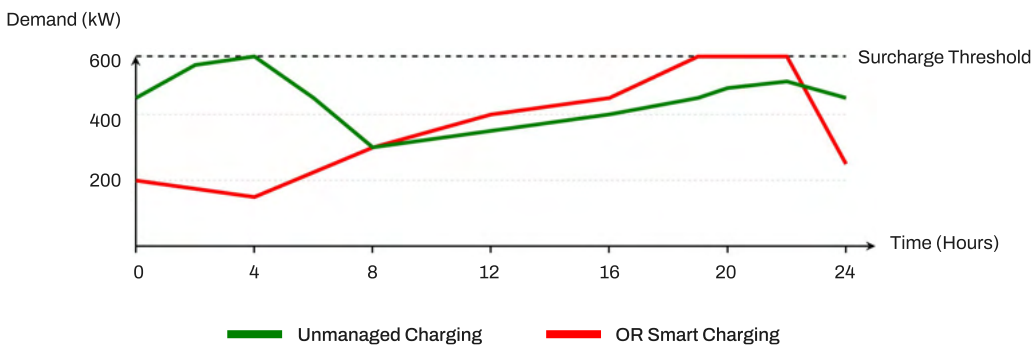


***Figure 3**: Impact of Charge Scheduling Optimization on depot power demand, highlighting peak shaving and shifting load to low-tariff nighttime windows.*

The Road Ahead: Hybrid OR-AI Frameworks

While exact mathematical optimization via commercial solvers (e.g., CPLEX, Gurobi) ensures rigorous optimality, large-scale networks with hundreds of buses introduce severe computational challenges. As the fleet size grows, the search space for monolithic MILP formulations expands exponentially, occasionally requiring hours or days to solve. This latency makes exact methods difficult to use for real-time dispatch adjustments.

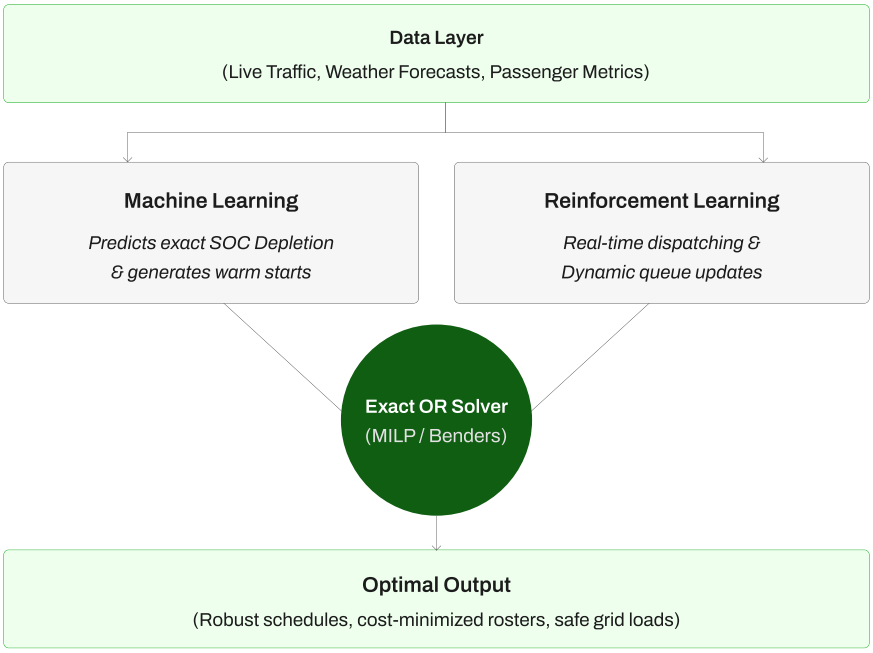


***Figure 4**: The modern hybrid planning architecture combining AI/ML predictive analytics with exact Operations Research optimization frameworks.*

To address this scale issue, recent transit literature highlights a shift toward hybrid architectures that pair Machine Learning (ML) with classical optimization frameworks [Fig. 4]:

- **Data-Driven Parameter Estimation**: Traditional OR models often treat consumption rates as static, historical averages. Modern ML methods learn from dynamic variables—including ambient temperature, real-time traffic data, and historical passenger loads—to predict state-of-charge (SOC) depletion profile along specific routes [4]. This yields high-fidelity parameters for downstream optimization.

- **Algorithmic Acceleration via Warm Starts**: ML classifiers can be trained to predict near-optimal initial solutions or baseline vehicle assignments. Passing these predictive ”warm starts” to exact decomposition algorithms (such as Benders Decomposition or Lagrangian Relaxation) significantly reduces the remaining search space, cutting compute times from hours to minutes.

- **Real-Time Receding Horizon Control:** Deep Reinforcement Learning (DRL) agents are being evaluated for online, real-time dispatch adjustments. In the event of an unexpected charger failure or unusual traffic delays, these trained policies adjust charge queue sequences instantaneously without requiring a full recalculation of the network-wide MILP from scratch.

While vehicle electrification is essential for reducing urban transport emissions, its operational success depends heavily on the underlying logistics. Combining exact mathematical programming with data-driven predictive models provides the scalable, robust decision-making architecture necessary to make large-scale electric bus transit viable.

References

[1] Council on Energy, Environment and Water (CEEW), India's Electric Vehicle Transition: Impact on Auto Industry and Grid, 2023.

[2] A. Ceder, Public Transit Planning and Operation: Theory, Modelling and Practice, CRC Press, 2007.

[3] X. Liu, A. Ceder, and X. Liang, "Transit timetabling coordination framework via mixed integer linear programming and metaheuristics," Transportation Research Part C, vol. 124, 2021.

[4] Global Transit Research Consortium, Machine Learning Applications in Electric Vehicle Scheduling and Energy Prediction, Journal of Sustainable Transportation, 2023.

[5] S. Bubeck, D. C. Ahuja, and T. S. Srinivas, "The electric vehicle scheduling problem: Models and solution methods," Transportation Research Part B: Methodological, vol. 93, pp. 112– 130, 2016.

[6] S. S. Perumal, R. M. Lusby, and J. Larsen, "Electric vehicle scheduling and crew scheduling: An integrated approach," Computers & Operations Research, vol. 132, p. 105300, 2021.

CHESS ENGINES

How to Make Machines “Learn”

Tarun Tiwari

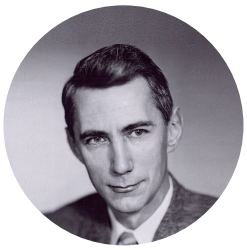
Pre-doctoral Research Associate—
Btech—Mtech IIT Kanpur

> *The implications go far beyond my beloved chessboard... Not only do these self-taught expert machines perform incredibly well, but we can actually learn from the new knowledge they produce.*
>
> — Garry Kasparov on AlphaZero

Chess engines are programs designed to find the best possible moves in a given chess position. Over the last seven decades, the development of chess engines has seen exponential growth thanks to software and hardware developments. This development reflects how the idea of making machines “learn” evolved since the beginning of the era of machine computing. This article traces that arc, how the underlying philosophy of developing chess engines shifted from telling machines what to think, to letting them figure it out themselves.

Shannon, Turing, and the Dawn of Chess Computing

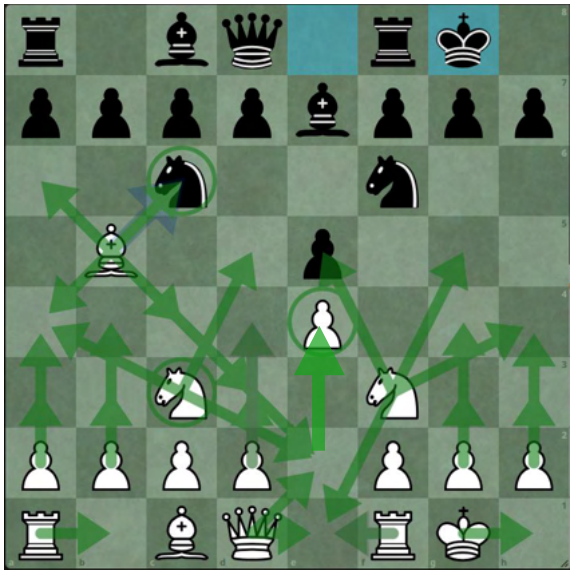
Alan Turing



Claude Shannon

To understand the current state of chess engines, we must start with pioneering works of Alan Turing and Claude Shannon in the late 1940s. Alan Turing along with David Champernowne came up with Turochamp [1,2], a chess program that computes moves. The algorithm was tested by computing the arithmetic by hand. Turochamp computes the best response by going through all possible moves of the player and then the opponent, i.e. 2 plies ahead (2 half moves) and stops there if there are no more checks, captures, recaptures or forced moves else it moves to higher depths. Once the tree ends, the positions reached are evaluated based on an evaluation function. The evaluation function was based on current pieces on the board with each piece given a value; also separate adjustments were made in the evaluation function based on piece mobility, piece defence and king safety.

Claude Shannon in 1950 wrote “Programming a computer for playing chess” [3] where he outlined the blueprint for chess engines for modern day computers. Like Turing, Shannon proposed searching 3-5 full moves ahead (or 6-10 plies or half moves). This idea of creating a search tree and finding moves by maximizing player’s evaluation is what is known as the minimax algorithm which is at the centre of all chess engines. Shannon also talks about how this is not enough, and one has to take care of a major capture happening just at or after the depth limit, which completely changes the evaluation. He suggested a ‘Quiescence’ search where to avoid evaluating a position at checks/captures, instead continue searching until board reaches a ‘quiet’ or stable state. Not just that, Shannon also added positional nuance into the evaluation function based on mobility, pawn structure and king safety. Now these are all heuristics which came from understanding of chess. The algorithm involved the use of a lot of human-learned understanding of chess. In fact, the basic piece value of queen being 9, rook being 5 also came from, as Shannon pointed out, “empirical evidence from various games”.



Possible legal chess moves in a position (White to move)

Early chess engines pioneered what is referred as heuristics in the computer science domain - providing “good enough” rules to the computer to get the work done.

The Minimax Algorithm

Chess is a two-player, zero-sum game. To evaluate a position, engines use a mathematical function that looks at a static board state and assigns it a numerical score. This score represents who is winning and by how much, without calculating any future moves. Traditionally, the setup is such that positive numbers mean White is winning (the maximizer), and negative numbers mean Black is winning (the minimizer). The number of positions to evaluate explodes as we keep going to higher depths. This is represented by what we call a game tree.[4]

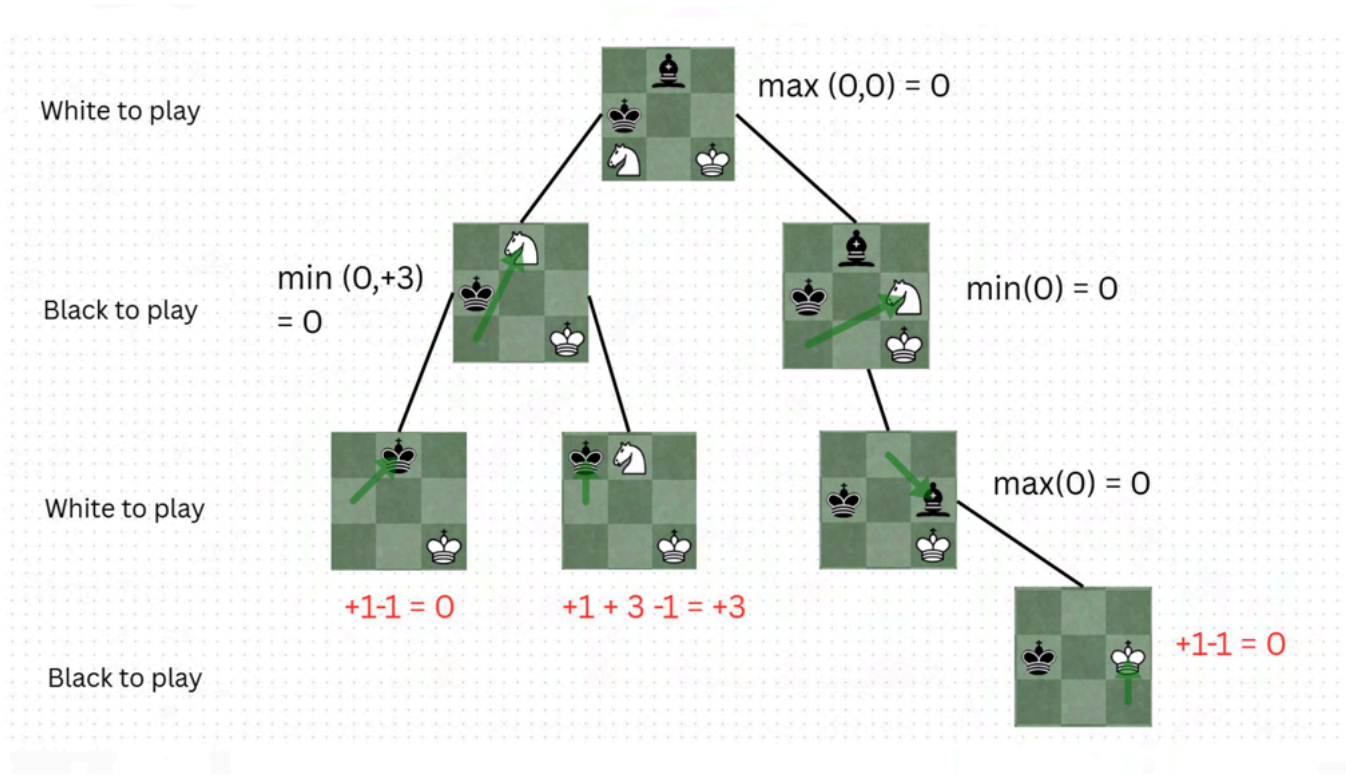


***Figure 1**: Here is a representation of the minimax algorithm on a game tree for a tiny chess game on a 3x3 board.*

The efficiency of the minimax algorithm is bound by its time complexity O (b^d), where b is the branching factor and d is the depth of the search. In a typical middlegame b ~ 35 and average human chess games last for 50 moves or 100 plies (half-moves). This complexity necessitated a compromise: evaluating a position up to a fixed depth and returning a static evaluation.

But the breadth here is still uncontrolled and grows fast. To control this, alpha-beta pruning is implemented. The core idea is to reject unnecessary evaluations by keeping track of the best possible evaluations using the variables alpha and beta for the two colors. Better ordering of positions to evaluate leads to more pruning. The time complexity with perfect ordering is O($b^{d/2}$).

Breaching the human threshold

The setup described so far for the chess engines remained the same for the next several decades. The changes came from better representation of chess positions on computers and small tweaks for better efficiency but largely the chess engines improved due to increased hardware capability. Moore's law took over and possible computations increased exponentially. From a few thousand computations to millions to billions per move, which led to major advances in accuracy.

The programs in the 80s were getting good enough to beat elite players, yet none could beat a grandmaster[5]. The first taste of success came when Deep Thought, which could search 700,000 positions per second, defeated grandmaster Bent Larsen. The watershed moment brewing up finally came in 1997 when the then World Champion Garry Kasparov was defeated in a match by IBM's Deep Blue computer 3.5-2.5. Deep Blue's computing power was driven by 32 processors. This hardware allowed the system to evaluate an impressive 200 million chess positions per second. Altogether, this equated to a processing speed of 11.4 billion floating-point operations per second [6].

1980

Elite-Level Chess Engines

— Beat elite players

— Not yet Grandmaster level

1988

Deep Thought

— 700K positions/second

— Defeated Bent Larsen

1997

Deep Blue

— Defeated Garry Kasparov

— 200M positions/sec

1997 marked the moment machines outperformed best human chess player, out-calculating 6-time World Champion Garry Kasparov

Deep Blue had 480 custom chess chip sets developed specially for computing chess positions. With so much computing power - it seemed like overkill to defeat a mere human computing at max 1 to 3 positions per second. But there is a catch, a grandmaster's search is precise - given a position the pruning is so good that only 3-5 possible moves remain to think about. The player's experience creates an intuition of where pieces belong. This is what was missing in chess engines, and all that computational effort was just compensating for not actually knowing what they were doing.

Chess engines started becoming useful for chess preparation at elite levels. Chess legends like Viswanathan Anand became early adopters of use of chess engines in professional chess preparation; many top-level players were in fact involved in developing various chess engines of the time — Fritz, Deep Blue, etc.— while Rybka, Stockfish, Komodo, and Houdini were late entrants.

Stockfish Era

Chess engines continued to improve with efficient code and better evaluation functions. More heuristics were added with the help of top players to better evaluate a position. But nothing substantial came in chess engine architecture for the next decade. Claude Shannon's idea was still relevant - it was just made more efficient and became more accurate due to increased computing power.

The availability of compute seemed to hold all the answers. IBM's BlueGene/P supercomputer was provided to Veselin Topalov for his World Championship match against Viswanathan Anand in 2010. The growth in computing power and the rise of GPUs around that time were setting the stage for a new era of machine learning.

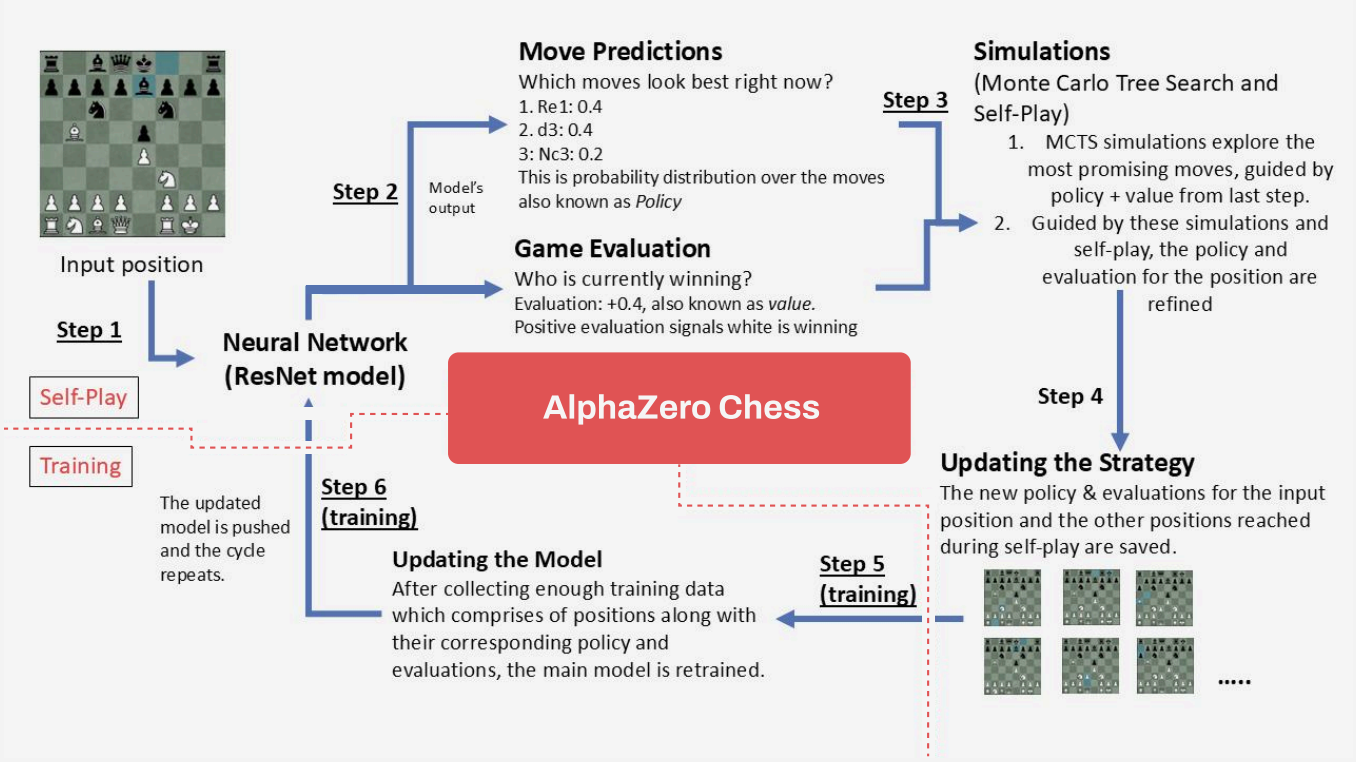
Stockfish came in 2008 and has been one of the leading chess engines since. Stockfish team in 2013 developed 'Fishtest'[7], a distributed testing framework that allows volunteers worldwide to donate CPU time to validate proposed code changes. When a developer submits an improvement, Fishtest distributes thousands of games between the current version and the candidate version across all contributing machines. The results are evaluated using the Sequential Probability Ratio Test (SPRT). SPRT is a statistical method that determines how many games to play to confidently accept or reject the hypothesis that the new version is stronger. It stops automatically once statistical significance is reached in either direction. This replaced ad-hoc human judgment with a rigorous, reproducible standard for what counts as a genuine improvement.

The AI Revolution: AlphaZero

The rise of neural networks fueled by exploding computing prowess, led Google Deep Mind in 2017 to implement neural nets in chess engines. This permanently altered the field of chess engines. AlphaZero discarded handcrafted evaluation entirely. It learned the game from scratch using self-play and reinforcement learning.

About two years before AlphaZero, Google DeepMind achieved something unprecedented in March of 2016. Their AlphaGo was able to defeat the Go World Champion Lee Sedol. The game of Go, unlike chess, wasn't conquered until 2016. The reason being the complexity of the huge number of possibilities in a position. The branching factor for Go is ~ 250 against 35 in chess.

AlphaGo [8] utilised popular ResNet architecture used in Image processing, for supervised learning on 30 million moves from human expert games. The network was trained to output a policy which is a probability distribution over possible legal moves. This trained network is used to generate data that will be used to train another network which takes in a position and outputs a single value which will be the evaluation of the position. For a given position, for the two networks' output - policy (probability distribution for legal moves) and value (evaluation) are computed, then a Monte Carlo Tree Search (MCTS)[9] runs thousands of quick simulations played until the end of the game considering the outputs from the networks and provide the final policy (final move) to be implemented.



How AlphaZero teaches itself chess.

This architecture was further improved when first AlphaGo Zero and then AlphaZero[10,11,12] was launched. AlphaZero has a single ResNet network that outputs both the policy and value from a given board position. The outputs are then utilised by MCTS to run simulations and update the policy. This new policy is then stored in a replay buffer. Many positions run through this process and the results are stored in a replay buffer. Concurrently, separate processors sample random batches from this buffer. They feed these positions into the neural network (the ResNet), compare its predictions against the actual recorded data, and tweak the network's weights to minimize errors, using the backpropagation method. This slightly smarter, updated network is broadcasted back to the self-play processors, creating a cycle of continuous self play.[13]

The AlphaZero neural network was trained for 9 hours, during which it played a staggering 44 million training games. The computational workload was supported by hardware consisting of a single machine equipped with 4 TPUs. It executes 800 Monte Carlo Tree Search (MCTS) simulations per move, which translates to a thinking time of approximately 40 milliseconds.

All this resulted in overwhelming success when put against the then best classical chess engine, Stockfish 8. In their initial 100-game match, AlphaZero remained completely undefeated, securing 28 wins and 72 draws, with 0 losses. In a series of later matches which had 12 batches of 100-games each with popular openings as the starting position, AlphaZero maintained its dominance by securing 290 wins and 886 draws, while conceding only 24 losses.

The defeat to AlphaZero and later to LeelaChess0 (Lc0)[14] (an open-source version of AlphaZero) in the TCEC (Top Chess Engine Championship) pushed Stockfish towards adopting neural nets in its engine. Stockfish in its version 12 released in 2020 adopted NNUE architecture (Efficiently Updatable Neural Networks)[15,16] which replaced earlier heuristics based evaluation function with the neural network based evaluation function. The Stockfish Elo jumped ~100 points with NNUE, which is a lot at that level. Since its release Stockfish has won all TCEC championships till date.

Conclusion

What makes the history of chess engines so remarkable is not just that machines got better at playing a game, but how their underlying methods evolved. The shift from hardcoded human rules to self-taught neural representations mirrors a broader transformation playing out across the entire field of artificial intelligence: we are moving from systems that only do what we tell them, to systems that can figure out the best approach on their own. AlphaZero didn't need a programmer to tell it that a bishop is worth three pawns; it discovered a richer, more profound understanding of the pieces entirely on its own.

Chess engines are not just being used to utilise heavy computing to get the best moves, engines like Maia learn to map human psychology, developing models that accurately emulate human play across various Elo ratings [17,18,19,20]. Researchers at Google Deep Mind in 2025 wrote a paper on generating creative chess puzzles using reinforcement learning [21]. Apart from finding the optimum move in a position, the chess researchers asking these questions signals something profound about where the field is heading.

Chess served as a laboratory, first for those who simply wanted to know the best move, and now for those asking whether machines can think creatively, surprise us, and even teach us things about our own cognition we hadn't considered. The questions have changed, but the board remains the same.

References

1. Chess Programming Wiki. (n.d.). Turochamp, from https://www.chessprogramming.org/Turochamp

2. Friedel, F., & Kasparov, G. (2017, September 23). Reconstructing Turing's "paper machine". ChessBase. https://en.chessbase.com/post/reconstructing-turing-s-paper-machine

3. Shannon, C. E. (1950). Programming a computer for playing chess. Philosophical Magazine, 41(314), 256-275. https://www.pi.infn.it/~carosi/chess/shannon.txt

4. UC Berkeley CS188. (n.d.-a). Minimax search. CS188: Introduction to Artificial Intelligence., from https://inst.eecs.berkeley.edu/~cs188/textbook/games/minimax.html

5. Friedel, F. (2015, June 6). Kasparov and thirty years of computer chess. ChessBase. Available at: https://en.chessbase.com/post/kasparov-and-thirty-years-of-computer-chess

6. IBM. (n.d.). Deep Blue - IBM History. https://www.ibm.com/history/deep-blue

7. Stockfish Developers. fishtest. GitHub repository. Available at: https://github.com/official-stockfish/fishtest

8. Silver, David, et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search." Nature, vol. 529, no. 7587, Jan. 2016, pp. 484–89, https://doi.org/10.1038/nature16961.

9. UC Berkeley CS188. (n.d.-b). Monte Carlo tree search. CS188: Introduction to Artificial Intelligence. Retrieved June 21, 2026, from https://inst.eecs.berkeley.edu/~cs188/textbook/games/monte-carlo.html

10. Google DeepMind. (2017). AlphaZero: Shedding new light on chess, shogi, and Go. https://deepmind.google/blog/alphazero-shedding-new-light-on-chess-shogi-and-go/

11. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815. https://arxiv.org/pdf/1712.01815

12. McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Hassabis, D., Kim, B., Paquet, U., & Kramnik, V. (2021). Acquisition of chess knowledge in AlphaZero. arXiv preprint arXiv:2111.09259. https://arxiv.org/pdf/2111.09259

13. OpenAI. (n.d.). Part 2: Kinds of RL algorithms. Spinning Up in Deep RL. Retrieved June 21, 2026, from https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

14. Leela Chess Zero. (n.d.). Neural network. Lc0 Developer Documentation. Retrieved June 21, 2026, from https://lczero.org/dev/old/nn/

15. Nasu, Yu. (2018). Efficiently Updatable Neural-Network-based Evaluation Function for Computer Shogi (NNUE). English translation. Available at GitHub repository: https://github.com/asdfjkl/nnue/blob/main/nnue_en.pdf

16. Stockfish Developers. (n.d.). NNUE. Stockfish. Retrieved June 23, 2026, from https://official-stockfish.github.io/docs/nnue-pytorch-wiki/docs/nnue.html

17. McIlroy-Young, R., Sen, S., Kleinberg, J., & Anderson, A. (2020). Aligning Superhuman AI with Human Behavior: Chess as a Model System. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20), 1677–1687. arXiv preprint arXiv:2006.01855. Available at: https://arxiv.org/abs/2006.01855

18. Tang, Z., Jiao, D., McIlroy-Young, R., Kleinberg, J., Sen, S., & Anderson, A. (2024). Maia-2: A Unified Model for Human-AI Alignment in Chess. Accepted at Advances in Neural Information Processing Systems (NeurIPS 2024). arXiv preprint arXiv:2409.20553. Available at: https://arxiv.org/abs/2409.20553

19. Monroe, D., Eilender, G., Chalmers, P., Tang, Z., & Anderson, A. (2026). Chessformer: A unified architecture for chess modeling. arXiv preprint arXiv:2605.19091. https://arxiv.org/pdf/2605.19091

20. CSSLab. maia-chess. GitHub repository. Available at: https://github.com/CSSLab/maia-chess

21. X. Feng et al., "Generating Creative Chess Puzzles," arXiv, Oct. 2025. doi: 10.48550/arXiv.2510.23881.

Academic Progression of Centre Researchers

Jepan Taji

Pre-doctoral Research Associate

M.Tech., Transportation Systems Engineering, IISc Bangalore

Joining University

University of Illinois Chicago

Program

PhD in Business Administration
(Information and Decision Sciences)

Shobhit Arora

Pre-doctoral Research Associate

B.Tech

Joining University

Carnegie Mellon University
Pittsburgh

Program

Master of Science in Computer Vision

CDSA x AI Summer Residency

The Centre for Data Science and Artificial Intelligence (CDSA), IIM Ahmedabad, was proud to contribute to the inaugural AI Summer Residency, a high-intensity, 4–6 week residential program jointly designed by IIMA Ventures and the Krishnamurthy Tandon School of Artificial Intelligence, IIM Ahmedabad.

The residency brought together 43 exceptional STEM students selected from over 11,000 completed applications, creating an immersive environment where participants worked on AI-first solutions for real-world challenges. Through structured mentorship, deep-work sprints, field immersion, and interdisciplinary learning, the program encouraged participants to move beyond prototypes and build AI solutions grounded in real user needs.

As part of the residency, the Centre supported an ecosystem of learning that connected participants with leading founders, researchers, faculty, and industry experts. The program featured sessions by **Akshay Akash**, Founder of **Stimular,** who shared insights from his entrepreneurial journey that began at IIM Ahmedabad; **Nirmam Sandesara**, Senior Director of Engineering at **Roblox**, who discussed engineering leadership and scaling AI-driven products; and **Naman Pushp**, Founder of **Airbound**, who shared perspectives on building deep-tech ventures.

Faculty sessions by **Prof. Ankur Sinha** challenged participants to identify emerging opportunities across sectors through case-based discussions, while Prof. **Amit Karna** led an interactive workshop on design thinking, helping teams refine problem statements and solution approaches.

To deepen their understanding of real-world applications, participants also visited the Institute for **Plasma Research (IPR)** and **Jaivel Aerospace**, gaining first-hand exposure to innovation in plasma science, aerospace engineering, and advanced technology. Throughout the residency, students explored AI applications across sectors including spacetech, enterprise AI, aerospace, plasma systems, and fintech.

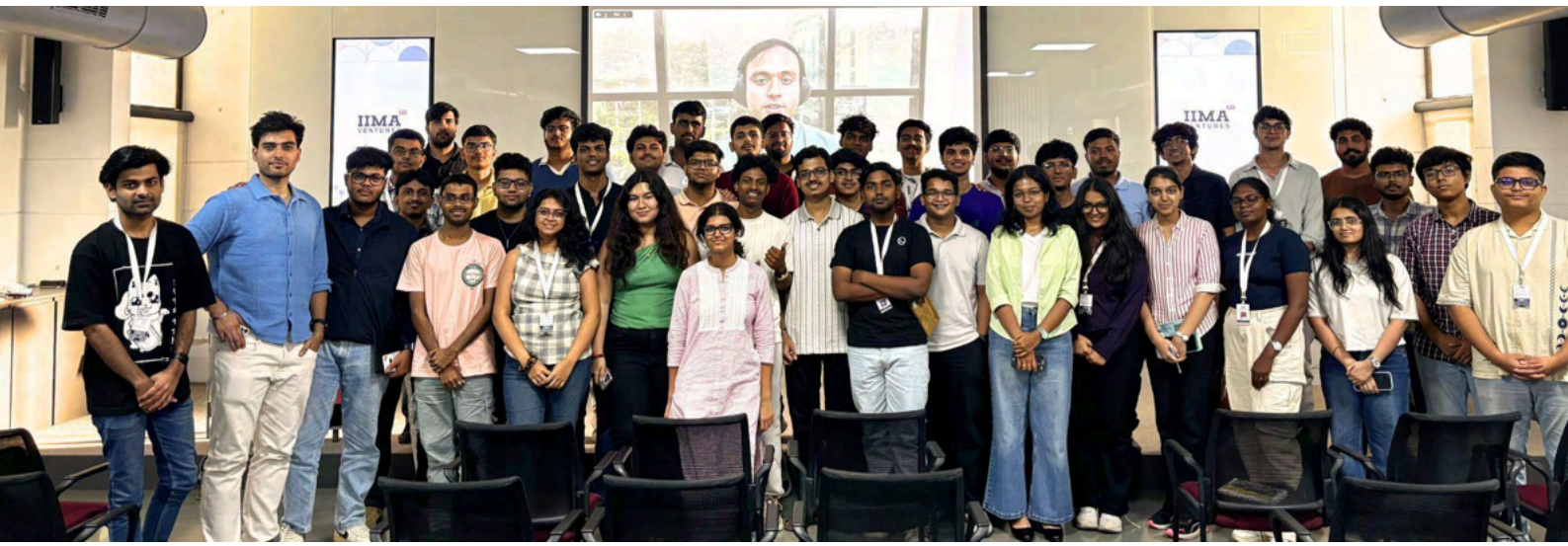
The final phase of the residency focused on field research, where teams engaged directly with users, operators, and stakeholders to validate assumptions and refine their ideas. Supported by mentorship clinics and one-on-one sessions with practitioners, participants learned to translate user insights into robust AI solutions while recognising that successful innovation begins with solving meaningful problems rather than applying technology for its own sake.

The Centre is proud to have been part of an initiative that reflects IIM Ahmedabad's commitment to advancing AI research, innovation, and entrepreneurship, while nurturing the next generation of AI builders capable of developing impactful, scalable, and responsible AI solutions.


Hi, I am Chhavie
My fags are Physics, Computation, Literature.


Stop
Year 1
Full Timepass Mode
Year 2
Germany Happened.
Year 3
The Leap.
Now
Somehow. Here.


Open Forum
Culture, Context & What to Expect
Ankur Sohanpal
IIMA Ventures
Manas Wats
IIMA Ventures
Adhwant Mishra
IIMA Ventures




Open Forum
Culture, Context & What to Expect
Ankur Sohanpal
IIMA Ventures
IIMA VENTURES

Upcoming Events

Brij Disa Centre for Data Science and Artificial Intelligence
IIMA INDIAN INSTITUTE of MANAGEMENT AHMEDABAD

X

ORSI Ahmedabad Chapter

CDSA - ORSI Competition on Practice of Management Science & Analytics

The Brij Disa Centre for Data Science and Artificial Intelligence (CDSA), Indian Institute of Management Ahmedabad, in collaboration with the Ahmedabad Chapter of the Operational Research Society of India (ORSI), is organizing the CDSA × ORSI Competition on Practice of Management Science and Analytics 2026–27.

The competition recognizes outstanding real-world applications of Operations Research, Management Science, and Analytics that have delivered measurable impact across business, government, and society. Inspired by the prestigious Franz Edelman Award of INFORMS (USA), it celebrates organizations that have successfully implemented analytical solutions to address complex challenges and improve decision-making.

Organizations operating in India are invited to submit projects demonstrating innovation, practical implementation, and measurable outcomes. Up to five finalist teams will be selected to present their work before an expert jury at IIM Ahmedabad, with one organization receiving the overall award.

Important Dates

30 Jun 2026	26 Jul 2026	05 Sep 2026	10 Jan 2027
Last Date for Entry	Selection of Semifinalists	Selection of Finalists	Final Presentations & Award

CDSA–ORSI Competition on the Practice of Management Science & Analytics 2025–26. Finalists receiving Certificates of Recognition.

Scan the QR code to learn more about the 2026–27 Award.

cdsa–orsi competition 2026–27

India Management Research Conference (IMRC) 2026

The India Management Research Conference (IMRC) 2026 will be held from 4–6 December 2026 at the Indian Institute of Management Ahmedabad, bringing together scholars, researchers, and practitioners to discuss emerging perspectives in management research. This year's conference is centred on the theme, "Reimagining Organizations: Transformation and Value-Creation in the Age of AI," reflecting the growing influence of artificial intelligence in reshaping organizations, decision-making, innovation, and value creation across industries.

The Brij Disa Centre for Data Science and Artificial Intelligence (CDSA), with the Krishnamurthy Tandon School of Artificial Intelligence, is organizing Track 10: Data Science & Artificial Intelligence. The track is jointly chaired by Prof. Samrat Gupta and Prof. Ankur Sinha.

The Centre welcomes original research on a broad range of topics, including AI-driven business models and organizational transformation; machine learning, generative AI, and advanced analytics; data-driven decision-making; AI applications across marketing, finance, operations, healthcare, and public policy; ethics, fairness, privacy, and responsible AI; and emerging frontiers such as AI for sustainability, platform ecosystems, data marketplaces, and AI in emerging markets.

Researchers are invited to submit their papers to Track 10 and contribute to advancing research at the intersection of artificial intelligence, data science, business, policy, and society.

Scan the QR code to learn more about Track 10 and submit your paper.

conference.iima.ac.in/imrc2026

Brij Disa Centre for Data Science and Artificial Intelligence

INDIAN INSTITUTE of MANAGEMENT AHMEDABAD

विद्याविनियोगाद्विकास:

INDIAN INSTITUTE of MANAGEMENT AHMEDABAD

Vastrapur, Ahmedabad 380015

www.iima.ac.in

+91 7971527514

cdsa@iima.ac.in

/Brij-Disa-Centre

Follow us on **LinkedIn**

Chairpersons

Ankur Sinha
(asinha@iima.ac.in)

Samrat Gupta
(samratg@iima.ac.in)

Centre Coordinator

Neaketa Chawla
(neaketac@iima.ac.in)

Centre Secretary

Jaydeep Gohel
(cdsa-secretary@iima.ac.in)