Review Article

Future trails for integer programming and relations to artificial intelligence

ABSTRACT

Aims: A review of methods and approaches for solving linear integer problems is presented in this work. These problems are classified as NP-hard optimization algorithms in artificial intelligence.
Study design:we have used the Google scholar to collect the data resources from past 5 years to analysis the techniques and methods used in different algorithms in artificial intelligence.
Methodology: Exact optimum solution for this class of challenges also need use of substantial computer resources. The current direction in which several researcher focuses their efforts to effectively address numerous difficult practical issues is the creation of efficient hybrid techniques that combine in an appropriate way the finest elements of multiple methods (precise or estimated). The approximation algorithms' core heuristic techniques might be classified as constructive algorithms and local-improvement algorithms.
Results: We examined three artificial intelligence algorithms utilizing the linear integer programming approach. Algorithm based on population It has also been demonstrated that a population of a critical size is necessary for a population-based optimization method to be effective. The genetic algorithm is shown next. The goal value associated with this solution may be utilized to effectively reduce the search tree in bound and branch type integer programming methods. Finally, we analyze the particle swarm optimization (PSO) approach, which demonstrates that In most cases, PSO outperforms the Branch and Bound method in solving such issues quickly.
Conclusion: Some initial computational findings are also shown to back up the idea that this technique deserves some consideration.

Keywords: linear integer programming, artificial intelligence, population based algorithm, genetic algorithm, PSO

1. INTRODUCTION

Integer programming (IP) related to artificial intelligence (AI) is projected to play an important part in the construction of AI-enabled technologies and services in the future. In particular, integer programming may improve the effectiveness and reliability of decision-making algorithms, as well as minimize the computation cost of AI-enabled operations. Furthermore, integer programming may be utilized to create AI-driven planning and routing solutions as well as to improve the precision of machine learning approaches. As AI becomes more powerful and prevalent, integer programming will play an increasingly important role in making AI-enabled systems more productive and accurate[1].

Despite its benefits, integer programming connected to artificial intelligence has significant disadvantages. One significant challenge is that IP is computationally expensive, necessitating vast volumes of data and processing. Furthermore, IP can be challenging to implement since it necessitates a substantial amount of expertise and knowledge to apply the approach to a specific situation. Moreover, because IP is based on addressing complicated optimization issues, developing methods that are simultaneously effective and efficient can be challenging. Finally, integer programming might provide difficult-to-interpret results, making it challenging to make judgments based on them[2].

Despite these disadvantages, IP in conjunction with artificial intelligence may be a valuable tool in the creation of AI-enabled apps and services.Organizations may acquire a better grasp of how AI-enabled technologies can provide them with increased decision-making capabilities, efficiency, and accuracy by showcasing the benefits of employing IP in AI-enabled applications[2].

A type of multipurpose constrained optimization model with integer variables, linear integer programming has an objective function that is a linear function and constraints that are linear inequalities. The optimization process using Linear Integer Programming (LIP) is illustrated in the following generalized example:

(1) Increase to max cx
(2) focus to: $Ax \leq b$,
(3) $x \in Z^n$,

Where $x \in Z^n$ is a vector of n integer variables: $x = (x1, x2... x_n)$. This formulation also includes equality constraints due to the fact that each equality constraint can be described by two inequality constraints. The data are rational and given by the mn matrix A, the 1n matrix c, and the m1 matrix b. in (2)[3].

Linear integer optimization problems can be used to describe ainclusive range of real-world issues in economics, politics, social science, logistics, and other fields. LIP problems can be used to solve combinatorial problems like, warehouse location, knapsack-capital budgeting , traveling salesperson, decreasing costs, machine and equipment selection, set covering areas, optimum flow challenges, corresponding challenges, weighted corresponding challenges, spanning trees challenges, and many planning problems. By simply approximating the required nonlinear functions using piece - wise functions, several nonlinear optimization problem with linear constraints might be converted into LIP optimization techniques[4].

2. MATERIAL AND METHODS / EXPERIMENTAL DETAILS / METHODOLOGY

Because IP optimization issues and LIP optimization approaches (1)-(3) belong to the category of NP-hard optimization algorithms, finding an optimal and even a workable result for large extentconditions is highly difficult and needs substantial computing effort. Finding an acceptable answer is usually more important than waiting a lengthy time for the ideal alternative. Some flexible limitations may remain in the issue model definition, and they can be adjusted only little. The precise algorithms must solve the issue even if one restriction is changed slightly. For real-world applications, this might be time-consuming and costly[5].The approximation methods are less sensitive to minor variations in some restrictions. Some of them tackle the problem sequentially, while others break it down into parts. In such cases, fixing the full problem is not required. As subroutines in precise algorithms, approximation algorithms have a wide range of applications. They might be used to find a suitable starting point, to reduce the available range of alternatives, or to guide the search for a perfect

answer. A wide range of approximation approaches have been devised to solve enormous real-world LIP optimization problems, with no guarantee that the final solution is optimum[6].

During the last thirty years, several local search-based Meta-heuristic algorithms have been created to circumvent the problem of local optimization and to reach a global optimum solution. They have been shown to be quite beneficial in practice. Many handbooks have been written in the last fifteen years devoted to the fundamental meta-heuristic techniques, their attributes and characteristics, as well as their usual applications. They are aimed at scientists, operations researchers, engineers, and application professionals who are seeking for the best optimization solution to handle specific challenges[7, 8].

The approximation algorithms may be split into three groups based on the accuracy of the results obtained[9]:

- Approximation algorithms with arbitrarily pre-determined accuracy (absolute or relative);
- Algorithms for approximation with a predetermined level of accuracy in which the forecast error is not typically zero.
- In this situation Heuristic algorithms, it is assumed, based on tests and other assessments, that they will discover a good quality solution to the issue using appropriate computer resources, but no quantitative evaluation of their correctness is provided.

Because it has been demonstrated analytically that they complete their work with a polynomial number of commonly used mathematical operations, it is necessary to develop approximate algorithms.

The following are the fundamental heuristic strategies used in approximation algorithms:

- constructive algorithms
- Local improvement algorithms.

Comment [SZBA1]: Only focus on 2 strategies

2.1 Constructive algorithms

The constructive algorithms generate the solution step by step using the issue data. The majority of the time, until the algorithm's efficiency is exhausted, there is no solution. This family includes the algorithm that is referred to as "greedy." In order to achieve the best local improvement, this algorithm adds a new step to the gained comprehensive.The traveling salesmen scenario is a well-known example of a greedy algorithm in action. Although these approaches are among the fastest approximation algorithms, they usually provide low-quality results. As a consequence, constructive algorithms usually include "look-ahead" procedures that analyze the future consequences and ramifications of the current decision[10, 11].

2.2 local-improvement algorithms

The idea behind local-improvement algorithms is simple. They start with an initial solution to the problem, often obtained through a constructive algorithm. The solutions evaluated are those in the vicinity of the current solution x, within N(x). If a better solution is found, it becomes the new current solution and its surrounding solutions are explored. This process continues until no more improvements can be made and the current solution becomes a local optimum. The method for determining the importance of a solution's neighborhood is clear. The set of solution that can be derived from x by using a simple transformation like tis referred to as the neighborhood N(x, t) of solutions x; consequently, diverse neighbors result from diverse transformations. There have been a number of proposals for finding a new best solution x [12]:

- a selection of x at random from N (x, t);
- (first-fit) the first solution that finds an improvement is used;
- All possible solutions around N(x, t) are looked into, and either the solution with the greatest improvement (best-fit) or some other intermediate condition is used.

The magnitude of this neighborhood is important because it shows how far away the recent solution is from creating a neighborhood where potential ideas can be looked at. The fact that these methods can only provide a local optimum is their primary flaw. Increasing the size of the neighborhood N(x,t) or attempting to describe various transition t that determine the neighborhood N(x, t), initial the search from random initial solutions selected at random in the possiblearea, attempting to perfect the search methods, and permitting the selection of a weak solution in the neighborhood N in some cases are some of the many approaches that can be taken to avoid this issue (x, t)[13].

The population-based algorithms, which are a comprehensive set of meta-heuristics based on common notions of the best survival and can learn,, are the most well-known and powerful meta-heuristics. Particle Swarm Optimization and Genetic Algorithms are two examples. The key heuristic techniques and algorithms are discussed briefly below.

Comment [SZBA2]: Is it part of the other 2 strategies?

3. RESULTS AND DISCUSSION

3.1 POPULATION BASED ALGORITHM

In contrast to the meta-heuristics that came before it, these [7] deal with a population, or group, of solutions. Moments of cooperation and self-adaptation mix at every iteration, which presents the typical search phase in the search space. Collaboration periods include the choosing, exchange, updating of information, or creation of test points through direct or indirect sharing of data collected during the search among individuals. Self-adaptation periods involve the application of a change, improvement in performance, or local search method. The theoretical foundations of population-based algorithms are depicted below:

Create afirst population of individuals;
While no stopping situation is met do
Co-operation,
Self-adaptation,
EndWhile

Optimization serves as a model for this approach to locating the feasible domain. The outcome of these methods is determined by how the group is modified. Population-based approaches for discrete optimization encompass evolutionary algorithms, ant colony optimization, and particle swarm optimization. To imitate natural evolution, a set of operations is applied to the current population in each generation to create individuals for the next one[14, 16]. In the population, the fitness value of each answer is determined. Some evaluations, whether experimental or otherwise, are used to convert it into an objective function or another objective function. Individuals with the greatest fitness levels are directly or indirectly engaged in the next population, generating new persons through a change or a combination of them. The operations applied are: alteration or mutation which directly changes individuals, and combination or crossover that creates new individuals through combining two or more individuals. The remaining solutions are discarded, indicating a decision has been made. Evolutionary algorithms use non-deterministic algorithms. They vary in how they generate, evaluate, select, and modify solutions. Besides genetic algorithms, the term "evolved algorithms" encompasses a broad range of other population-

based algorithms as almost all functions can be freely generated and modified to address specific problems[17].

Algorithm for population based algorithm using linear programming

Input: Problem parameters N; (w_j ,r_j), for j = 1, 2,...,N**;**
W_i for i = 1, 2...; objective target η; population size n; parent size μ
Output: Number of rows (H); chosen solution matrix x_{ij} for
i=1,2,….H and j= 1,2,….N; gained objective f^T

3.1.1 Outcome of the study

The proposed population-based method demonstrates the effectiveness of the linear programming approach by using a combination of techniques, such as tournament selection to pick the best parent solutions, recombination to merge desirable traits from two parents, and mutations to improve offspring solutions. With each iteration, solutions improve exponentially fast and the desired target can be reached within 16 to 18 iterations for a problem with a million variables.

Such a big optimization issue has rarely been attempted to be addressed in practice, especially with such few solution evaluation and such a short computing time.The major grounds for its effectiveness are the exploitation of the problem's linearity in constructing a recombine operators throughout a collection of solutions and their subsequent repair procedures.

The ability of a population-based optimization algorithm's recombine operator to combine great and partial data from two or more members of the population into a single new solution is its strength. The recombination operator is primarily to blame for the convincing demonstration of the population-based linear programming method's effectiveness.

3.2 GENETIC ALGORITHM

Comment [SZBA3]: May consider to move to LR section and include only result/findings in this section

In the mid-1960s, few pioneering works on Genetic algorithms (GA) were presented; they apply many methods for boosting search efficiency [18, 19]. Their method mimics species genetic evolution. Individuals are a population of alternative GA solutions. A selection operator is used to randomly select a number of parent-child pairs from the current population. Each pair reproduces through a crossover operator, resulting in the birth of two new individuals (solution) known as offspring.During natural evolution, a mutation operator is utilized to randomly change the progeny individuals replicating the mutation with a low likelihood. Individuals from the population with lower objective function values are eventually replaced by their children with higher objective function values. This technique is repeatedly performed until the population no longer improves or after a set number of repetitions (generations). Despite the fact that GA has been shown to be useful for a wide range of issues, there is no assurance that it will result in the most effective solution. Their convergence can be affected by the genetic operators used, the likelihood of mutation, and the selection criteria used, all of which necessitate fine-tuning of the parameters. Despite the fact that there is a fundamental explanation for GA's effectiveness, most circumstances do not easily fall into this model. It is probable to combine GA with other heuristic strategies and organize them as parallel algorithms[20].

3.2.1 Application to Branch and Bound algorithm

In this part, we suggest using GA to discover a good first feasible integer solution to a LIP.

$$\min c^T x, \quad (A)$$
$$\text{subject to } Ax = b, \quad (B)$$
$$l \le x \le u, \quad (C)$$
$$x_j \text{ integer}, j = 1, \ldots, p \le n, \quad (D)$$

Where $A \in R^{m \times n}$ and all vectors are of appropriate dimension.

In our example, a gene is a feasible integer value of an integer variable that meets (C), and a genome is a viable vector that meets (C). If p n, there is a vector with more than 0 non-integer elements. The objective is to create and use a GA to find a (hopefully) great integer-solvable problem. Because only the values of the objective function will be used, it is not necessary for the solutions to be comparable[21].

Algorithm for GA for LP

1. Work through the relaxed LP challenge. Set k=1, S=ϕ. Form anfirst population P1, then choose a genome from the population P_k.
2. Fix the genes and determine the genome's value.
3. If the answer meets (2) and (3), proceed to 4, otherwise proceed to 5.
4. If the answer is satisfactory, save it in S.
5. If there are any remaining genomes in the population, proceed to step 2, otherwise proceed to step 6.
6. If k = K, go to B&B; otherwise, go to 7.
7. Add the population Pk to the set of stored possible solutions S; Pk: = Pk S.
8. k:= k + 1. Crossover, mutation, and selection activities are used to determine the next generation Pk, goto 2.

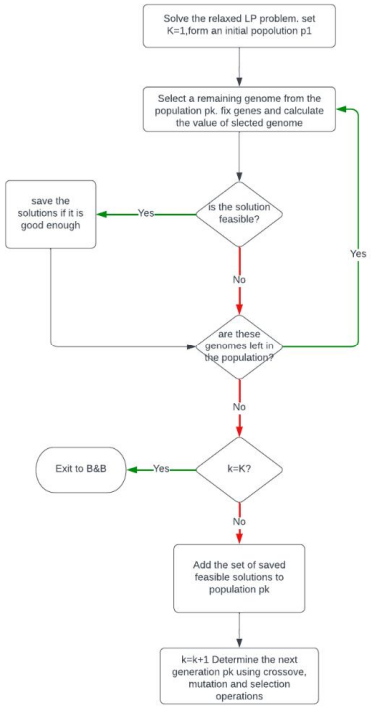

Figure 1: Block diagram for finding a good feasible integer solution by GA.

3.2.2 Outcome of the study

When the Branch and Bound (B&B) procedure begins, the matched objective value is utilized as the dominant solution if the GA gives an integer viable solution. Wide branches of the trees can be excluded from the search if they are adequate, contributing to the B&B's overall efficiency.

3.3 PARTICLE SWARM OPTIMIZATION (PSO) algorithm

The goal of PSO is to replicate the collective behavior of schools of fish, bee swarms, bird flocks, and other living things. Each particle in PSO moves to a new location by utilizing three vectors - collaboration, competition, and friction. The velocity of the particle, v(t), is considered to calculate the inertia vector, which is weighted by a constant factor w. This results in a representation of the particle's inclination to keep its current velocity [22, 23]. The particle y's (tpresent) position is linked to its best individual place in the competition vector during the search phase. A randomly distributed function is used to weight this vector. The collaboration vector connects the particle's present position y(t) to the particle's global best

position. A second randomly distributed function is used to weight this vector. It is obvious that particle collaboration is critical for determining the global best solution. For the particles to avoid entrapment in the local minima, inertia and competition are required[24].

3.3.1 Branch and bound technique

The BB method can be algorithmically drawn as follows [25]

1. Begin with a relaxed feasible region $M_0 \supset S$ and partition M_0 into finitely many subsets M_i ; i = 1; 2;:::;m, where S is the feasible area of the problem.
2. For each subset M_i , determine lower (and if possible) upper bounds, $\beta(Mi)$ and $\alpha(Mi)$, respectively, satisfying
   $\beta(M_i) \leq \inf f(M_i \cap S) \leq \alpha(M_i)$ where f is the objective function under concern.
   Then, the bound defined as
   $\beta := \min_{i=1,2,3....,m} \beta(M_i)$
   And
   $\alpha := \min_{i=1,2,3....,m} \alpha(M_i)$
   Are the complete bound i.e.
   $\beta \leq \min f(s) \leq \alpha$
3. If $\alpha = \beta$ ($\alpha - \beta \leq \varepsilon$ for pre-defined constant $\varepsilon > 0$), then stop.
4. If not, select some of the subsets Mi and partition them to create a more defined M0 partition. Repetition of the procedure is required to establish new, hopefully, improved bounds for the new partition elements.

Outcome of the study

The benefit of the BB method is that it usually lets you reject subclasses of S where the least of *f* cannot be reached during the iteration process. The BB method encounters significant obstacles, such as correctly separating the viable zone and selecting the sub problem to investigate. Integer Programming issues have been solved successfully using the BB method. The fundamental integer problem is transformed into a continuous one by the approach presented in this paper. Then, it uses Sequential Quadratic Programming to solve the sub-problems that result from limiting the parameters' scope, which is still considered continuous. This method is repeated until all of the variables have integer values.

4. CONCLUSION

In reality, a wide range of real-world problems are described by integer optimization problems. Their size and population are always increasing. Even though exact algorithms for integer problems have improved significantly in recent years, their excessive runtimes and memory requirements typically render them useless for actual medium and large-scale problems. Theoretical and computational research has been published on multiple or binary problems. Therefore, problems with 0-1 variables or unique structures benefit from the most common heuristic methods for obtaining excellent initial solutions, which are included in the commercial programming software. In general, solving the integer problem is still significantly more difficult.

Because achieving a decent viable answer in a reasonable period is totally adequate for many real issues, developing heuristic algorithms with polynomial computing cost remains a

current topic. Because of their exponential computing complexity, many large-scale real-world issues cannot be handled by precise algorithms. In this circumstance, the only option is to employ approximation polynomial time methods.

Competing interests

Declaration of competing interest should be placed here. All authormust disclose any financial and personal relationships with other people or organizations that could inappropriately influence (bias) their work. Authorhave declared that no competing interests exist.

COMPETING INTERESTS DISCLAIMER:

Authors have declared that they have no known competing financial interests OR non-financial interests OR personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] H. Xiao, B. Muthu, and S. N. Kadry, "Artificial intelligence with robotics for advanced manufacturing industry using robot-assisted mixed-integer programming model," *Intelligent Service Robotics,* pp. 1-10, 2020.

[2] H. Liang, M. Tsuei, N. Abbott, and F. You, "AI framework with computational box counting and Integer programming removes quantization error in fractal dimension analysis of optical images," *Chemical Engineering Journal,* vol. 446, p. 137058, 2022.

[3] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt, "A survey on mixed-integer programming techniques in bilevel optimization," *EURO Journal on Computational Optimization,* vol. 9, p. 100007, 2021.

[4] C. G. Valicka, D. Garcia, A. Staid, J.-P. Watson, G. Hackebeil, S. Rathinam*, et al.*, "Mixed-integer programming models for optimal constellation scheduling given cloud cover uncertainty," *European Journal of Operational Research,* vol. 275, pp. 431-445, 2019.

[5] Y. Liang and G. Cheng, "Topology optimization via sequential integer programming and canonical relaxation algorithm," *Computer Methods in Applied Mechanics and Engineering,* vol. 348, pp. 64-96, 2019.

[6] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," *arXiv preprint arXiv:2006.10518,* 2020.

[7] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," *Advances in neural information processing systems,* vol. 32, 2019.

[8] M. Huisman, J. N. Van Rijn, and A. Plaat, "A survey of deep meta-learning," *Artificial Intelligence Review,* vol. 54, pp. 4483-4541, 2021.

[9] S. Barman and S. K. Krishnamurthy, "Approximation algorithms for maximin fair division," *ACM Transactions on Economics and Computation (TEAC),* vol. 8, pp. 1-28, 2020.

[10] Y. He, Y. Chen, J. Lu, C. Chen, and G. Wu, "Scheduling multiple agile earth observation satellites with an edge computing framework and a constructive heuristic algorithm," *Journal of Systems Architecture,* vol. 95, pp. 55-66, 2019.

[11] U. J. Mele, L. M. Gambardella, and R. Montemanni, "A new constructive heuristic driven by machine learning for the traveling salesman problem," *Algorithms,* vol. 14, p. 267, 2021.

[12] M. H. Shirvani, "A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems," *Engineering Applications of Artificial Intelligence,* vol. 90, p. 103501, 2020.
[13] H. Wang and B. Alidaee, "Effective heuristic for large-scale unrelated parallel machines scheduling problems," *Omega,* vol. 83, pp. 261-274, 2019.
[14] A. Slowik and H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," *Neural Computing and Applications,* vol. 32, pp. 12363-12379, 2020.
[15] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, "An improved differential evolution algorithm and its application in optimization problem," *Soft Computing,* vol. 25, pp. 5277-5298, 2021.
[16] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation,* vol. 24, pp. 380-393, 2019.
[17] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms–A survey," *Swarm and evolutionary computation,* vol. 44, pp. 695-711, 2019.
[18] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications,* pp. 43-55, 2019.
[19] A. Lambora, K. Gupta, and K. Chopra, "Genetic algorithm-A literature review," in *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*, 2019, pp. 380-384.
[20] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications,* vol. 80, pp. 8091-8126, 2021.
[21] O. Ben-Ammar, P. Castagliola, A. Dolgui, and F. Hnaien, "A hybrid genetic algorithm for a multilevel assembly replenishment planning problem with stochastic lead times," *Computers & Industrial Engineering,* vol. 149, p. 106794, 2020.
[22] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access,* vol. 10, pp. 10031-10061, 2022.
[23] E. H. Houssein, A. G. Gad, K. Hussain, and P. N. Suganthan, "Major advances in particle swarm optimization: theory, analysis, and application," *Swarm and Evolutionary Computation,* vol. 63, p. 100868, 2021.
[24] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in particle swarm optimization," *Swarm and Evolutionary Computation,* vol. 58, p. 100718, 2020.
[25] Y. Ren, Z. Lu, and X. Liu, "A branch-and-bound embedded genetic algorithm for resource-constrained project scheduling problem with resource transfer time of aircraft moving assembly line," *Optimization Letters,* vol. 14, pp. 2161-2195, 2020.