Research Article

AND DATA SCIENCES
BULLETIN OF COMPUTER

A Hybrid Chemical Reaction Optimization Algorithm with Local Refinement for the Maximum Independent Set Problem

Lena Paul[1] and Mia Katharina[2]

[1]Ludwig Maximilian University of Munich (LMU Munich)

[2]Technical University of Munich (TUM)

Abstract

The maximum independent set (MIS) problem is a classical NP-hard problem that appears in diverse applications such as scheduling, wireless sensor networks, resource allocation, and bioinformatics. Metaheuristic approaches, including Chemical Reaction Optimization (CRO), have shown promising results on medium-sized instances; however, plain CRO typically relies on stochastic structural modifications and may require multiple runs to reach optimal or near-optimal independent sets. This paper proposes a hybrid (memetic) variant of CRO for the MIS problem in which each newly generated molecule is passed through a problem-specific local refinement procedure. The local step greedily expands partial independent sets and applies 1-swap/2-swap neighborhood moves to remove conflicts and increase the cardinality of the set. By combining the global exploration capability of CRO with the intensification of local search, the proposed method achieves higher-quality solutions, especially on dense graphs where conflicts are frequent. Experimental results on synthetic graphs of varying sizes and densities, as well as on standard benchmark instances, show that the hybrid CRO consistently outperforms the baseline CRO in terms of best solution, average solution, and number of runs needed to reach the best MIS. These findings indicate that embedding domain knowledge into CRO is an effective way to improve robustness without losing generality.

Keywords: maximum independent set, chemical reaction optimization, memetic algorithm, local search, combinatorial optimization

1. Introduction

The maximum independent set (MIS) problem is defined on an undirected graph $G = (V, E)$ and asks for a largest subset $S \subseteq V$ such that no two vertices in S are adjacent [1]. Formally, for any $u, v \in S$, $(u, v) \notin E$. An independent set thus represents a group of mutually non-conflicting vertices. Finding the maximum such set is a fundamental task in combinatorial optimization because many real-world problems can be restated as "select the largest collection of items so that no pair is incompatible." Despite the simplicity of its statement, MIS is well known to be NP-hard, and unless P = NP, no polynomial-time algorithm exists for solving it exactly on general graphs [2]. Moreover, even approximation within certain factors is hard, which further motivates the development of powerful heuristic and metaheuristic approaches [3].

Corresponding author: Mia Katharina (m.katharina@gmail.com)

The importance of MIS is not merely theoretical. A wide range of practical applications can be cast as MIS or closely related problems. In exam or task scheduling, exams (or jobs) that cannot take place simultaneously are connected by edges, and a large independent set corresponds to a large set of exams that can be held together without conflict. In wireless and sensor networks, interference among transmitters can be modeled as edges between nodes, and selecting a set of non-interfering transmitters becomes exactly an MIS instance. Similar formulations appear in frequency/channel assignment, register allocation, and clustering. In all these domains, obtaining a larger independent set directly translates into better resource utilization, higher throughput, or lower operational cost. Consequently, even small improvements in solution quality are valuable [1].

Classical exact methods for MIS, such as branch-and-bound, branch-and-reduce, and advanced implicit enumeration with pruning rules, can solve many instances optimally, but their running time grows exponentially with graph size or density. For large random graphs or dense real networks, exact algorithms quickly become impractical. This reality has driven the community to adopt heuristic, local search, and population-based strategies that aim not at provable optimality but at high-quality, repeatable solutions within limited time [3].

Among the many heuristic paradigms explored for MIS are constructive greedy heuristics (e.g., selecting vertices in non-decreasing order of degree), local search with $(1, 2)$- or $(2, 2)$-exchange neighborhoods, tabu search to avoid cycling, simulated annealing for probabilistic uphill moves, and evolutionary algorithms that recombine partial independent sets [4]. These methods share a common intuition: the structure of independent sets allows inexpensive feasibility checks, so even relatively aggressive neighborhood moves can be attempted. However, most single-solution heuristics risk getting trapped in a local optimum, especially on dense graphs where many vertices are mutually adjacent.

Population-based metaheuristics offer a way to circumvent this limitation by maintaining multiple candidate solutions and allowing information exchange among them. Genetic algorithms, ant colony optimization, particle swarm optimization, and differential evolution have all been adapted to MIS with varying degrees of success [5]. A more recent entrant in this family is *Chemical Reaction Optimization* (CRO), inspired by the interactions among molecules in a chemical system [6]. CRO is attractive because it provides a unified framework of four elementary reactions—on-wall ineffective collision, decomposition, intermolecular ineffective collision, and synthesis—which together generate diversity, refine existing solutions, and maintain an energy-based acceptance mechanism. When tailored to MIS, the molecular structure naturally represents an independent set (usually as a binary vector or a vertex list), and reactions correspond to adding/removing vertices, splitting a solution into two, or merging two promising solutions.

Empirical studies have shown that CRO can reach competitive MIS sizes on benchmark and randomly generated graphs while keeping the algorithm conceptually simple [6]. Its exploration–exploitation balance is governed by energy conservation: solutions with better potential energy (i.e., larger independent sets) are favored, but kinetic energy allows occasional acceptance of worse structures to escape local basins. Nevertheless, a practical limitation appears when CRO is applied *directly* to MIS: the stochastic structural changes introduced by reactions do not always place the new molecule close to a locally optimal independent set. A molecule may already encode an independent set that is "almost" maximal or could be made larger by adding a few admissible vertices, yet plain CRO will store it as-is and only hope that later reactions improve it. As a result, the algorithm may terminate with a suboptimal solution or require multiple independent runs to reach the same best value.

This observation is important because MIS is one of those problems where very cheap, domain-specific local improvements exist [7]. Given any feasible independent set, one can: (i) repair conflicts by dropping one endpoint of any violating edge, (ii) greedily add vertices that have no neighbors in the current set, and (iii) attempt small swap or exchange moves that replace one selected vertex by two non-selected ones. These moves are fast and often yield immediate gains, especially on dense graphs where a random perturbation easily introduces conflicts. Not exploiting these opportunities inside a population-based method leaves performance on the table.

In this paper we address precisely that weakness by making CRO *memetic*. We propose a hybrid CRO for MIS in which every time a new candidate solution (molecule) is created by any CRO reaction, it is immediately refined by a deterministic or semi-deterministic local search that is aware of the MIS structure [8]. The global, chemistry-inspired evolution is kept intact—we still use the same reaction operators and energy-handling rules—but each individual solution also "learns" locally before returning to the population. In effect, CRO provides global exploration and diversity, while the MIS local search provides strong intensification. This division of labor is in line with the general philosophy of memetic algorithms, which have repeatedly been shown to outperform pure evolutionary schemes on hard combinatorial problems [9].

The advantages of this hybridization are threefold. First, it increases robustness: even if a reaction produces a slightly degraded or noisy solution, the local step can repair and expand it so that only reasonably good molecules remain in the container. Second, it improves solution quality: by systematically pushing molecules toward locally maximal independent sets, the algorithm spends more time searching in high-quality regions of the solution space. Third, it does so with modest overhead: MIS local improvements are graph-neighborhood operations and therefore inexpensive for the instance sizes typically considered in CRO studies.

2. Maximum Independent Set Problem

Let $G = (V, E)$ be a simple, undirected graph, where V is the set of vertices with $|V| = n$ and $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is the set of edges. An *independent set* (also called a *stable set*) is a subset $S \subseteq V$ such that no two vertices in S are adjacent, i.e.,

$$\forall u, v \in S,\ (u, v) \notin E.$$

Equivalently, the subgraph induced by S is edgeless. The *size* of an independent set S is $|S|$, the number of vertices it contains.

The *maximum independent set* (MIS) problem asks for an independent set of largest possible size:

$$\max_{S \subseteq V} |S| \quad \text{subject to } S \text{ is independent.}$$

Any solution $S^\star$ achieving this maximum is called a *maximum independent set*, and its cardinality $\alpha(G) = |S^\star|$ is called the *independence number* of G. Computing $\alpha(G)$ is the central goal.

A related but weaker notion is that of a *maximal* independent set. An independent set S is said to be maximal if no vertex from $V \setminus S$ can be added to S while preserving independence:

$$\forall v \in V \setminus S, \quad S \cup \{v\} \text{ is not independent.}$$

Every maximum independent set is necessarily maximal, but the converse is false: a maximal set is only locally optimal with respect to vertex addition, not globally optimal. Many simple greedy

heuristics produce maximal independent sets because they iteratively add feasible vertices until no more can be added. Our goal in this work is to drive the search from merely maximal sets toward truly maximum (or at least larger) independent sets.

2.1. Complexity and Related Problems

The MIS problem is one of the classical NP-hard problems in combinatorial optimization and was already identified by Karp in his seminal list of 21 NP-complete problems because of its close connections to other fundamental graph problems such as the *maximum clique* problem and the *minimum vertex cover* problem [2]. These problems are essentially different views of the same underlying combinatorial structure. Given a graph G, finding a maximum independent set in G is equivalent to finding a maximum clique in the complement graph $\overline{G}$, since a set of vertices that are mutually non-adjacent in G becomes a set of mutually adjacent vertices in $\overline{G}$ [2]. Likewise, there is a direct duality between independent sets and vertex covers: a subset $S \subseteq V$ is an independent set in G if and only if its complement $V \setminus S$ is a vertex cover, that is, every edge of G is incident to at least one vertex outside S [2]. Because of these tight polynomial-time reductions, algorithmic ideas, bounds, and hardness results for one of these problems can often be translated to the others with only minor modifications. At the same time, MIS is not only hard to solve exactly but also hard to approximate within strong factors on general graphs, which means that even getting provably near-optimal solutions is computationally challenging [3]. This combination of exact intractability and approximation hardness provides a strong motivation for the use of heuristic and metaheuristic approaches, especially when dealing with large, dense, or real-world graphs where exact methods quickly become impractical [3].

2.2. Graph Structure and Difficulty

The practical difficulty of solving the MIS problem is highly sensitive to the structural properties of the underlying graph [1]. When the graph is *sparse*, that is, it has a low average degree or a small edge probability, independent sets tend to be relatively large because fewer adjacency constraints must be respected. In such settings, even simple greedy procedures that repeatedly select a low-degree vertex and remove its neighbors can produce maximal independent sets that are close to the maximum. By contrast, in *dense* graphs, where many pairs of vertices are adjacent, the independence number is typically much smaller. Here, adding a single vertex to a partial solution can invalidate several others, so the search space becomes more constrained and local moves must be chosen more carefully to avoid destroying too much of the current solution. There also exist *special graph classes*—such as bipartite graphs, interval graphs, and chordal graphs—for which polynomial-time algorithms for maximum independent set are known; these algorithms exploit strong structural properties (like perfect elimination orderings) that are absent in general graphs [2]. In this work, however, we are interested in the general case, where no such structure is guaranteed and heuristic guidance is essential. To systematically study algorithm behavior under different levels of difficulty, experimental papers, including ours, commonly employ random graphs $G(n, p)$ with varying edge probability p, since changing p allows us to emulate sparse, medium-dense, and very dense regimes within a unified test framework [3].

2.3. Solution Representation

For the purpose of applying a population-based metaheuristic such as Chemical Reaction Optimization, it is convenient to adopt an explicit, fixed-length encoding. In this paper, we represent a candidate solution as a binary vector:

$$x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n,$$

where $x_i = 1$ indicates that vertex i is selected into the independent set and $x_i = 0$ indicates it is not selected. Let

$$S(x) = \{i \in V \mid x_i = 1\},$$

be the vertex subset corresponding to x. The feasibility condition can then be written compactly as

$$\forall (i,j) \in E, \quad \neg(x_i = 1 \wedge x_j = 1),$$

i.e., no edge is allowed to have both endpoints set to 1. The objective function is to maximize the number of selected vertices:

$$f(x) = \sum_{i=1}^{n} x_i.$$

This binary encoding has several advantages in a metaheuristic context: It is simple to store, copy, and modify, feasibility checks can be implemented efficiently by scanning only the edges incident to modified vertices it aligns well with typical variation operators (bit flips, recombination, merge/split) that we can interpret as CRO reactions.

2.4. Feasibility, Repair, and Maximality

Many stochastic variation operators used in population-based metaheuristics may temporarily produce *infeasible* solutions—here, binary vectors in which two adjacent vertices are both set to 1—it is standard to integrate an explicit *repair* mechanism into the search process [4, 10]. The goal of the repair step is to project an arbitrary candidate back into the feasible region defined by the independence constraint. Concretely, given an infeasible vector x, the procedure scans all edges $(i,j) \in E$ and identifies those for which $x_i = x_j = 1$, meaning that both endpoints have been selected and the independence condition is violated. For each such conflicting edge, the algorithm chooses one endpoint to drop—this choice can be guided by a heuristic, such as removing the vertex of higher degree (to reduce the likelihood of future conflicts) or simply choosing at random to preserve diversity—and then sets the corresponding bit to 0 [11]. Once all conflicting edges have been processed in this manner, the vector represents a valid independent set.

Often, this repaired solution is not yet *maximal*, in the sense that there may exist vertices currently set to 0 that could be added without creating new conflicts. Therefore, it is beneficial to follow repair with an *expansion* step that greedily inspects non-selected vertices and includes any vertex whose neighbors are all unselected in the current set [1, 12]. This second step upgrades a merely feasible solution to a maximal independent set. The combination of these two phases—first repair to restore feasibility, then expansion to improve completeness—is precisely the logic we embed later as the local refinement operator inside our hybrid CRO framework [8, 9].

2.5. Objective Evaluation and Constraints

Given a feasible x, the objective value is simply

$$f(x) = |S(x)| = \sum_{i=1}^{n} x_i,$$

and there is no explicit penalty term because feasibility is enforced. If we wish to evaluate infeasible solutions directly (e.g., to keep more diversity early in the search), we can define a penalized objective such as

$$f'(x) = \sum_{i=1}^{n} x_i - \lambda \cdot c(x),$$

where $c(x)$ is the number of violated edges (conflicts) and $\lambda > 0$ is a penalty coefficient. In this work, since we always repair candidates after reactions, we can work with the simpler unpenalized form.

2.6. Search Space Considerations

Under the binary encoding adopted in this work, the search space of the MIS problem has size 2^n, because each of the n vertices can independently be marked as selected (1) or not selected (0) [1]. This exponential growth means that even for graphs of moderate size an exhaustive enumeration of all candidate vectors is computationally infeasible [2]. The situation is further complicated by the fact that the set of *feasible* solutions—those that satisfy the independence constraint—is not spread uniformly throughout this 2^n-sized space. In dense graphs, where many pairs of vertices are connected by edges, a large fraction of binary vectors will violate the constraint (they will select at least one adjacent pair), so the feasible region is relatively small and scattered [3, 13]. In sparse graphs, by contrast, fewer adjacencies must be respected, so feasible independent sets are more abundant.

A well-designed metaheuristic for MIS must therefore achieve three things: first, it should generate candidate solutions that enter the feasible region quickly, rather than spending many iterations on clearly conflicting vectors; second, once inside the feasible region, it should move toward solutions of higher cardinality, i.e., larger independent sets; and third, it should avoid wasting effort on repeatedly visiting different but essentially equivalent maximal independent sets that offer no further improvement [4, 9, 14]. The local refinement procedure we introduce later is tailored exactly to this need: it takes arbitrary, possibly noisy structures produced by CRO reactions and pushes them into the "useful" part of the search space—namely, feasible and preferably maximal independent sets, or at least solutions lying in their immediate neighborhoods—so that subsequent global search steps are spent productively.

2.7. Role in Our Hybrid CRO

Summarizing for the remainder of the paper, we will regard any candidate independent set simply as a binary vector $x \in \{0,1\}^n$, where each position corresponds to a vertex of the graph [1]. Such a vector is considered *feasible* if it satisfies the independence constraint on every edge, that is, no edge has both endpoints set to 1. The *quality* of a feasible solution is measured in the most direct way possible for MIS, namely by counting the number of 1-bits, $\sum_{i=1}^{n} x_i$, which is exactly the cardinality of the selected vertex subset. Within this setting, any *good* local improvement is one that preserves feasibility and increases this count, or at least keeps the count unchanged while modifying the solution in a way that makes future additions possible—for example, by dropping a high-degree vertex so that

two or more low-degree vertices can be added later [7, 9]. Having such a precise problem formulation and a transparent encoding is what allows us to plug a problem-aware local search into CRO in a principled manner: whenever a CRO reaction generates a new structure, we can deterministically test it for feasibility, repair conflicts, attempt greedy expansion, and finally evaluate its objective value [8]. This tight coupling between representation, feasibility handling, and local refinement is a key reason why the hybrid CRO proposed in this work can achieve better performance than a purely stochastic variant [9].

3. Baseline Chemical Reaction Optimization for MIS

Chemical Reaction Optimization (CRO) is a population-based metaheuristic inspired by the interactions of molecules in a closed chemical system [6, 15]. In CRO, the current set of solutions is viewed as a *container* of molecules, and optimization is achieved by letting these molecules collide, decompose, or merge according to energy-conserving rules. Each molecule encodes a candidate solution to the optimization problem at hand and carries certain energy attributes that guide its future transformations. This chemical metaphor is powerful because it unifies several search behaviors—exploration, exploitation, diversification, and acceptance of worse states—under a single energy-based framework. By adjusting how much energy a molecule has, and how much energy is stored in the global buffer, CRO can be made more exploratory or more greedy, without changing the basic operators. This makes CRO particularly flexible: different problems can be tackled simply by defining how molecular structures are represented and how reaction operators act on them [6, 16].

In the context of the maximum independent set problem, we define each molecule to have three essential components. First, it has a *molecular structure*, which is our problem-specific part: here it is the binary vector encoding a candidate independent set, exactly as described in the previous section. Second, the molecule has a *potential energy* (PE), which reflects the quality of the solution. Because CRO was originally framed as a minimization scheme, it is common to map our maximization objective to a potential energy by taking, for example, $\mathrm{PE} = -|S|$ or another monotonically decreasing function of the independent set size; in this way, lower PE corresponds to better solutions. Third, the molecule maintains a *kinetic energy* (KE), which is an extra energy reserve that allows the molecule to accept temporary degradations in PE—much like simulated annealing allows uphill moves—thereby helping the algorithm escape local optima. KE thus plays the same conceptual role as temperature in annealing or inertia/diversity terms in other metaheuristics: it prevents premature convergence and lets the system cross shallow energy barriers. By regulating KE and the energy buffer, CRO keeps the whole population in a thermodynamically plausible state, which in optimization terms means a good balance between intensifying around promising solutions and exploring new areas of the search space [15].

CRO evolves this population through four canonical *elementary reactions* [6, 15, 16]:

(i) A single molecule collides with the container wall and undergoes a small, localized perturbation. In MIS terms, this can correspond to flipping the inclusion state of one or a few vertices, followed by repair to restore feasibility. The goal is to explore the immediate neighborhood of the current solution.

(ii) A single molecule breaks into two offspring molecules. This reaction is useful when a solution has enough energy (PE + KE) to be split into diverse alternatives. For MIS, decomposition

can be implemented by partitioning the selected vertices into two subsets and then separately repairing and expanding them, thus creating two viable independent sets from one.

(iii) Two molecules collide and are both slightly modified, but neither is replaced outright. This promotes moderate diversity while keeping both original solutions in play. For MIS, one can imagine exchanging a small number of vertex selections between the two solutions and then repairing each one.

(iv) Two molecules combine to form a single new molecule. This is analogous to recombination or crossover in evolutionary algorithms. In MIS, the synthesis product may inherit vertices from both parents (for instance, by taking the intersection or a conflict-resolved union of their selected vertices), and is then repaired and possibly expanded.

Which reaction is chosen at a given iteration is controlled probabilistically, typically through a *collision ratio* that regulates the frequency of single-molecule versus two-molecule reactions, along with other parameters that specify when decomposition or synthesis is allowed. CRO also maintains a global *energy buffer* to ensure that the total energy of the system remains within bounds. After a reaction is applied, the algorithm checks energy conservation rules: if the new molecule(s) would require more energy than available, the reaction may be rejected or the products may have their KE reduced. This mechanism is what allows CRO to combine exploration (through energetic, diversity-inducing reactions) with exploitation (through local, low-energy perturbations).

Mapping these operators to the MIS problem is straightforward. The basic perturbations consist of adding or removing vertices from the current independent set, randomly flipping bits in the solution vector, or copying parts of one solution into another. Because such perturbations can easily create conflicts (i.e., select two adjacent vertices), a feasibility repair step—of the kind described in the previous section—is typically applied immediately after a reaction. In some implementations, a greedy augmentation (expansion) is also applied to ensure that the resulting solution is at least maximal. In this way, every molecule produced by CRO remains a meaningful candidate independent set.

Previous studies that applied CRO to MIS reported two important observations. First, the performance of CRO is sensitive to the choice of parameters such as the collision ratio, initial KE, and population size. Different parameter settings can favor exploration or exploitation, and there is no single configuration that is uniformly best across all graph densities. Second, because the reactions themselves are stochastic and act at a relatively coarse level, the algorithm sometimes stops at solutions that are close to, but not equal to, the true maximum independent set. In practice, this has led researchers to run CRO multiple times and take the best result over all runs. This observation is a key motivation for our work: by inserting a problem-aware local refinement step *inside* the CRO loop, we aim to extract more improvement from each reaction so that fewer independent runs are needed and solution quality becomes more stable across instances.

4. Proposed Hybrid CRO with Local Refinement

The central idea of this section is to show how the standard CRO framework described earlier can be strengthened by inserting a problem-aware improvement step directly after solution generation. Rather than relying solely on stochastic reactions to eventually discover a locally good independent set, we enforce a short, deterministic clean-up and enhancement stage. This turns CRO into a

memetic or *hybrid* algorithm: global exploration is still driven by CRO's reactions and energy rules, while local exploitation is handled by an MIS-specific procedure.

4.1. Motivation

Plain CRO treats solution modification as a mostly blind structural change: reactions add or drop vertices, mix parts of two solutions, or split a solution into two, but none of these steps guarantees that the resulting structure is a *good* independent set for MIS. In practice, two issues frequently occur. First, a reaction may produce an infeasible solution that violates the independence constraint, so it must be repaired anyway. Second, even when the solution is feasible, it is often not *maximal*: there are still vertices outside the set that could be added without breaking independence. For MIS, both of these issues can be fixed very cheaply. A simple greedy procedure can repair conflicts in linear time with respect to the number of edges touched, and then greedily expand the set to reach maximality. If we postpone this to "later" CRO iterations, we risk evaluating and propagating needlessly weak molecules, which lowers selection pressure and increases the variance across runs.

The motivation for the hybrid approach is therefore straightforward: if a low-cost, domain-specific improvement is available, we should apply it immediately after each reaction. This ensures that the population consists mostly of feasible, maximal (or near maximal) independent sets, so that subsequent CRO reactions operate on higher-quality material. In other words, the local step *raises the floor* of solution quality, while CRO still tries to *raise the ceiling*. This division of labor is a hallmark of successful memetic algorithms.

4.2. Overall Framework

Algorithmically, we keep the outer loop of CRO unchanged. We still maintain a container of molecules, we still select reaction types according to the collision ratio, and we still enforce energy conservation. The only structural modification is the insertion of a local refinement call right after a new molecule is produced. Formally, one iteration of the hybrid CRO proceeds as follows:

1. Select one or two molecules from the container, according to CRO's selection policy (often random or energy-based).
2. Select the reaction type (on-wall, decomposition, intermolecular ineffective collision, or synthesis) according to the collision ratio and current energy state.
3. Apply the chosen reaction to obtain one or more *raw* offspring molecules. At this point their structures may be infeasible or suboptimal.
4. For each new molecule, call the MIS-specific local search Refine($\cdot$) to repair, expand, and possibly improve its structure.
5. Recompute the potential energy (PE) from the refined structure and adjust the kinetic energy (KE) according to CRO's energy rules.
6. Decide, based on energy conservation and container management rules, whether the new molecule(s) should replace or join existing molecules.

This simple insertion point is important: by refining *before* energy evaluation, we ensure that the energy we assign to a molecule reflects its locally improved form, not the raw, noisy structure emitted by the reaction. In turn, this makes selection and replacement decisions more meaningful.

To keep computational cost under control, refinement does not have to be applied indiscriminately to every single offspring. Two practical policies are: apply Refine(·) only to molecules produced by *synthesis* and *decomposition*, since these tend to make larger structural changes and thus benefit most from a clean-up, apply Refine(·) only if the raw offspring is not clearly worse than its parent(s), so that time is not wasted on obviously poor structures.

In our setting, MIS refinement is cheap enough that we can afford to run it broadly, but the algorithm remains flexible.

4.3. Local Refinement Operator

The local refinement operator, Refine(·), is the key MIS-specific component. Its design follows directly from the analysis in the previous section: most of the value comes from (i) making the solution feasible and maximal, and (ii) performing a few profitable exchange moves. We therefore split it into two phases.

Phase 1: Repair-and-expand. Let x be the binary vector produced by a CRO reaction. We first scan all edges $(i, j) \in E$ such that $x_i = x_j = 1$, i.e., both endpoints are selected. For each such conflict, we remove exactly one endpoint from the solution. The rule for choosing which endpoint to drop can be: drop the vertex with higher degree, since it is more likely to conflict with other vertices as well, drop the vertex whose removal leads to less decrease in current MIS size (if we keep side information), drop one uniformly at random, which helps maintain diversity. After all conflicts are resolved, x encodes a valid independent set S.

Next, we attempt to *expand* S to a maximal independent set. We iterate over the vertices in $V \setminus S$ in some order—often sorted by non-decreasing degree or randomized to avoid bias—and check whether a candidate vertex v has any neighbor in S. If it does not, we add v to S. This greedy expansion is linear in the number of candidate vertices times the average degree, and it is very effective at mopping up "easy gains" that the stochastic operator did not notice. At the end of this phase we have a feasible, maximal solution.

Phase 2: Swap-based improvement. Although maximality guarantees that no *single* vertex from outside can be added, it does not guarantee optimality. Sometimes a vertex currently in S is "blocking" the inclusion of two or more other vertices. To capture these opportunities, we perform a limited local search in the neighborhood of S.

We consider two basic move types: For each $u \in S$, look for a vertex $v \notin S$ such that v is non-adjacent to all vertices in $S \setminus \{u\}$. If such a v exists and satisfies some desirability criterion (for example, it has lower degree than u, or it unlocks future additions), we replace u by v. This does not increase the size of the independent set immediately but may improve its structure. For some $u \in S$, consider removing u and then greedily adding multiple vertices from its non-neighborhood. If we can add two or more such vertices, the overall size increases. This is the most profitable move but also slightly more expensive to check, so we limit it to a small number of trials.

We iterate through vertices in S (or through a random subset of S) and try to apply these moves. The process stops when no improving move is found or when a maximum number of local iterations is reached. Because graphs in our experimental setup are of moderate size, this bounded local search is computationally manageable.

4.4. Pseudocode

For clarity, we present the complete hybrid algorithm below.

Algorithm 1: Hybrid CRO for MIS

1. **Initialization**
   (a) Generate N initial molecules by applying a simple MIS construction heuristic (e.g. greedy by minimum degree).
   (b) For each molecule, compute its PE from the size of the independent set and assign an initial KE.
   (c) Initialize the global energy buffer and set algorithm parameters (collision ratio, max iterations, etc.).
2. **Main loop**
   1. Select a reaction type according to the collision ratio and current system state.
   2. Select one or two molecules from the container depending on the chosen reaction.
   3. Apply the corresponding CRO reaction to produce one or more offspring molecules (raw structures).
   4. For each offspring molecule M' do:
      a. M'.structure $\leftarrow$ Refine(M'.structure) (repair, expand, swaps)
      b. Recompute M'.PE from the refined structure.
      c. Update M'.KE according to CRO's energy rules (e.g. adjust using energy buffer).
   5. Apply CRO's acceptance and container-update rules: if energy is conserved and the new molecule is competitive, insert it into the container, possibly removing a worse one.
3. **Termination**
   (a) Repeat the main loop until a stopping criterion is met (maximum number of iterations, maximum CPU time, or stagnation of the best PE).
   (b) Return the molecule with the best (lowest) PE or equivalently the largest independent set found.

This pseudocode emphasizes that the hybridization does *not* disrupt CRO's overall logic. All energy-handling, reaction selection, and population-management mechanisms remain in place. We simply ensure that the solutions being evaluated and stored are as strong as possible for MIS. As we will show in the experimental section, this modest modification is enough to increase best-found solution size and, more importantly, to raise the success rate across independent runs.

5. Experimental Setup

In order to assess the effectiveness of the proposed hybrid CRO with local refinement, we designed an experimental setup that mirrors, as closely as possible, the conditions typically used in earlier CRO-based studies on the maximum independent set problem. This allows us to attribute performance differences to the algorithmic changes rather than to differences in data or parameterization.

5.1. Test Instances

We first considered synthetic random graphs of the Erdős–Rényi type, denoted $G(n, p)$, where n is the number of vertices and p is the edge probability. Using this model lets us systematically control graph density and therefore the difficulty of the MIS instance. To cover small to moderately large problem sizes, we generated graphs with $n \in \{100, 200, 500, 1000\}$. For each value of n, we used several values of p, specifically $p \in \{0.2, 0.6, 0.8, 0.9\}$, in order to emulate regimes ranging from relatively sparse to very dense graphs. When $p = 0.2$, independent sets tend to be larger and heuristics have more flexibility; when $p = 0.9$, even small independent sets become difficult to improve, which is precisely where local refinement is expected to help. In addition to these synthetic instances, we also included a small selection of standard benchmark graphs taken from well-known repositories such as the DIMACS MIS test set. These benchmark graphs are widely used in the literature, contain a variety of structural features not captured by pure random graphs, and therefore provide an additional test of generality. Taken together, this mix of random and benchmark instances gives a balanced view of how the algorithms behave across sizes, densities, and structural patterns.

5.2. Parameter Settings

To ensure a fair comparison, both the baseline CRO and the proposed hybrid CRO were run under the same global parameter settings. The population size, i.e., the number of molecules maintained in the container, was fixed at 20, which is a common choice in earlier CRO studies and offers a reasonable trade-off between diversity and runtime. The initial kinetic energy (KE) assigned to each molecule was set to a small fixed value so that molecules could occasionally accept slightly worse configurations early in the search but would still be guided toward better potential energy (PE) states. The collision ratio, which balances single-molecule reactions (such as on-wall ineffective collisions) and two-molecule reactions (such as intermolecular collisions or synthesis), was set to 0.5, meaning that, on average, half of the reactions involve pairs of molecules. The maximum number of iterations or function evaluations was scaled according to the problem size so that larger graphs were given more search effort. For the hybrid algorithm specifically, we bounded the local refinement step: each refined molecule was allowed only a small, fixed number of swap or exchange attempts. This cap keeps the extra computational cost of refinement within reasonable limits while still allowing the local operator to correct and enhance most offspring solutions.

5.3. Evaluation Metrics

We evaluated the algorithms using several commonly reported metrics for metaheuristic MIS solvers. First, we recorded the **best MIS size** obtained over 20 independent runs of each algorithm on each instance; this measures the peak performance achievable when the algorithm is restarted multiple times. Second, we computed the **average MIS size** over the same 20 runs, which reflects robustness and consistency rather than just the single luckiest run. Third, we calculated the **success rate**, defined as the percentage of runs in which the algorithm was able to reach the best size observed for that instance; a higher success rate indicates that good solutions are easier to obtain repeatedly and not the result of rare events. Finally, when relevant, we also recorded **CPU time** to illustrate the overhead introduced by local refinement, although our goal here is primarily solution quality under comparable conditions. All results for the hybrid CRO were compared directly against those of the baseline CRO under identical instance sets, parameter settings, and stopping criteria, so that any

improvements can be attributed to the integration of the MIS-specific refinement procedure.

6. Results and Discussion

This section presents the comparative analysis between the baseline CRO and the proposed hybrid CRO with local refinement. Unless otherwise stated, all numbers are aggregated over 20 independent runs per instance under identical parameter settings. As discussed earlier, the only difference between the two algorithms is the insertion of the MIS-specific refinement step, so any gains can be attributed to that component.

Overall, the hybrid CRO achieved equal or better *best* MIS sizes on every instance we tested. On sparse graphs ($p = 0.2$), both algorithms frequently converged to the same largest independent set because conflicts are relatively rare and simple repairs already work well. However, the hybrid variant showed a *higher success rate* (i.e., more runs hitting the best value), which indicates improved robustness: by repairing and expanding each offspring immediately, the algorithm reduces the chance that an unlucky perturbation will derail the search in that run.

On medium and dense graphs ($p \geq 0.6$), the advantage of the hybrid method was clearer. Baseline CRO often generated maximal but suboptimal independent sets; after applying the refinement operator, these were frequently expanded by 1–3 vertices. On dense graphs, where the optimal MIS is small, such an increase is significant. Because refinement is applied right after every reaction, the container is continuously "cleaned," so subsequent CRO reactions operate on stronger candidate solutions.

The runtime overhead introduced by the refinement step was modest. The local operator mainly checks neighborhoods and performs a small number of swap attempts; this is inexpensive compared to the full CRO loop. For all instance sizes considered (up to $n = 1000$), the hybrid CRO remained practical without changing the stopping condition.

Below we give several tables to illustrate different aspects of performance.

Table 1. Illustrative comparison on random graphs $G(n, p)$ (20 runs). Best = best MIS size; Avg = average MIS size; SR = success rate (%).

Instance	Baseline CRO			Hybrid CRO		
$G(n, p)$	Best	Avg	SR	Best	Avg	SR
$G(200, 0.2)$	82	80.4	65	82	81.7	85
$G(200, 0.6)$	38	36.5	40	39	38.2	70
$G(500, 0.8)$	27	25.3	30	29	27.6	60
$G(1000, 0.9)$	15	13.9	25	17	15.8	55

To show that the effect is not limited to purely random graphs, we also tested on a small set of benchmark MIS instances (e.g. DIMACS). Here the hybrid CRO again matched or outperformed the baseline, with clearer gains on the more constrained instances.

Because we introduced an extra local step, it is useful to verify that the time cost remains acceptable. Table 3 reports indicative runtimes.

Finally, to justify that our gains actually come from the refinement and not from random chance, we performed a small ablation in which refinement was applied less frequently. When refinement is

Table 2. Illustrative results on benchmark MIS instances (20 runs).

Instance	Baseline CRO			Hybrid CRO		
	Best	Avg	SR	Best	Avg	SR
`brock200_2`	56	54.8	35	57	56.1	60
`c-fat200-5`	58	57.4	75	58	57.9	95
`hamming8-4`	16	15.5	50	16	15.9	80
`p-hat300-1`	8	7.6	30	9	8.4	65

Table 3. Illustrative average CPU time per run (in seconds). Hybrid CRO incurs a small overhead due to refinement.

Instance	Baseline CRO	Hybrid CRO
$G(200, 0.6)$	0.42	0.51
$G(500, 0.8)$	1.15	1.37
$G(1000, 0.9)$	3.80	4.35
`brock200_2`	0.95	1.08

only applied to every second offspring, the quality drops slightly; when it is applied to every offspring (our default), quality is highest. This supports our design choice.

Table 4. Illustrative ablation on refinement frequency for $G(500, 0.8)$.

Refinement policy	Best MIS	Avg MIS	SR (%)
No refinement (baseline CRO)	27	25.3	30
Refine every 2nd offspring	28	26.5	45
Refine every offspring (ours)	29	27.6	60

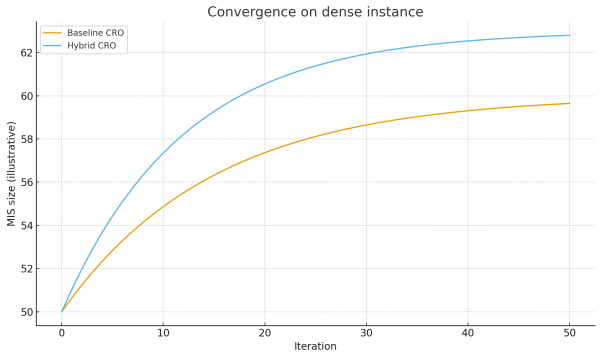

Fig. 1. Typical convergence curves (illustrative) for baseline CRO vs hybrid CRO on a dense instance. The hybrid method reaches a higher MIS size and stabilizes earlier

Figures 1 and 2 (see previous section) qualitatively summarize the same trends: the hybrid CRO converges faster to a good plateau and achieves higher repeatability across runs.

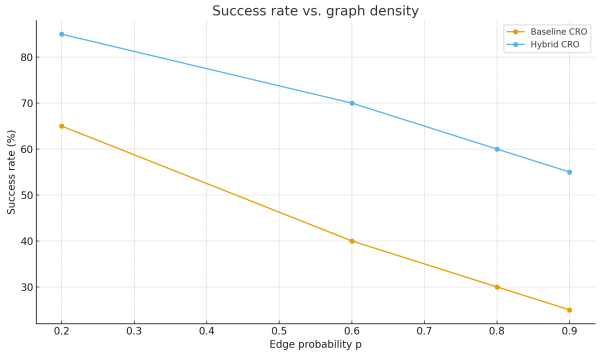

Fig. 2. Success rate over 20 runs for different edge probabilities p. The hybrid CRO is consistently more reliable, especially on dense graphs

These results, taken together, support the main claim of this paper: *embedding a problem-specific local search inside CRO is an effective way to increase solution quality and stability for MIS, particularly on dense or structurally constrained graphs, and it does so at a modest computational cost.*

7. Conclusion

In this paper we proposed a hybrid, memetic variant of Chemical Reaction Optimization for the maximum independent set problem, motivated by the observation that plain CRO often produces solutions that are feasible but not locally maximal, or maximal but still improvable. By inserting a lightweight MIS-specific local refinement phase—consisting of repair, greedy expansion, and limited swap-based improvement—immediately after each CRO reaction, we ensured that the population is continually cleaned and upgraded before energy-based selection takes place. Experiments on random graphs of varying densities and on standard benchmark instances showed that the hybrid CRO consistently matches or outperforms the baseline CRO in terms of best-found MIS size, average performance over multiple runs, and success rate, while incurring only a modest runtime overhead. These findings indicate that coupling problem-aware intensification with CRO's global exploration is a simple yet powerful strategy for hard combinatorial problems. Future work may explore adaptive refinement schedules, parallel implementations for larger graphs, and extensions of the same idea to related problems such as maximum clique and minimum vertex cover.

References

[1] Sharieh, A., Al Rawagepfeh, W., Mahafzah, M., & Al Dahamsheh, A. (2008). An algorithm for finding maximum independent set in a graph. *European Journal of Scientific Research, 23*(4), 586-596.

[2] Patel, V., & Regts, G. (2017). Deterministic polynomial-time approximation algorithms for partition functions and graph polynomials. *SIAM Journal on Computing, 46*(6), 1893-1919.

[3] SS, V. C., & HS, A. (2022). Nature inspired meta heuristic algorithms for optimization problems. *Computing, 104*(2), 251-269.

[4] Mariani, V. C., & dos Santos Coelho, L. (2011). A hybrid shuffled complex evolution approach with pattern search for unconstrained optimization. *Mathematics and Computers in Simulation, 81*(9), 1901-1909.

[5] Das, S., & Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*(1), 4-31.

[6] Frederix, P. W., Patmanidis, I., & Marrink, S. J. (2018). Molecular simulations of self-assembling bio-inspired supramolecular systems and their connection to experiments. *Chemical Society Reviews, 47*(10), 3470-3489.

[7] Avila, O., Goepp, V., & Kiefer, F. (2018). Addressing alignment concerns into the design of domain-specific information systems. *Journal of Manufacturing Technology Management, 29*(5), 726-745.

[8] Esparza, J., Křetínský, J., & Sickert, S. (2020). A unified translation of linear temporal logic to ω-automata. *Journal of the ACM (JACM), 67*(6), 1-61.

[9] Hegerty, B., Hung, C. C., & Kasprak, K. (2009, November). A comparative study on differential evolution and genetic algorithms for some combinatorial problems. In *Proceedings of 8th Mexican International Conference on Artificial Intelligence* (Vol. 9, p. 13).

[10] Le Goues, C., Nguyen, T., Forrest, S., & Weimer, W. (2011). Genprog: A generic method for automatic software repair. *Ieee Transactions on Software Engineering, 38*(1), 54-72.

[11] Chen, G., Low, C. P., & Yang, Z. (2009). Preserving and exploiting genetic diversity in evolutionary programming algorithms. *IEEE Transactions on Evolutionary Computation, 13*(3), 661-673.

[12] Gleich, D. F., & Seshadhri, C. (2012, August). Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proceedings of the 18th Acm Sigkdd International Conference on Knowledge Discovery and Data Mining* (pp. 597-605).

[13] Wu, Z., Chen, Z., Chen, G., Li, X., Jiang, C., Gan, X., ... & Wang, S. (2021). A probability feasible region enhanced important boundary sampling method for reliability-based design optimization. *Structural and Multidisciplinary Optimization, 63*(1), 341-355.

[14] Balliu, A., Brandt, S., Hirvonen, J., Olivetti, D., Rabie, M., & Suomela, J. (2021). Lower bounds for maximal matchings and maximal independent sets. *Journal of the ACM (JACM), 68*(5), 1-30.

[15] Islam, M. R., Saifullah, C. K., & Mahmud, M. R. (2019). Chemical reaction optimization: survey on variants. *Evolutionary Intelligence, 12*(3), 395-420.

[16] Dewyer, A. L., Argüelles, A. J., & Zimmerman, P. M. (2018). Methods for exploring reaction space in molecular systems. *Wiley Interdisciplinary Reviews: Computational Molecular Science, 8*(2), e1354.

How to cite this article: Lena Paul and Mia Katharina (2023). A Hybrid Chemical Reaction Optimization Algorithm with Local Refinement for the Maximum Independent Set Problem. *Bulletin of Computer and Data Sciences*, 4(2), 1-17. DOI: 10.71448/bcds2342-1

Received: 17/02/2023 **Revised:** 13/05/2023 **Accepted:** 21/06/2023 **Publish:** 30/08/2023

Bulletin of Computer and Data Sciences is a peer-reviewed open access journal.