CIRRELT

CIRRELT-2024-21

Parallel Meta-Heuristic Search

Teodor Gabriel Crainic

July 2024

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Parallel Meta-Heuristic Search

Teodor Gabriel Crainic*

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and School of Management, Université du Québec à Montréal

Abstract. This chapter presents and unified and up-to-date overview of the parallel meta-heuristics field. It synthesizes, classifies, and describes the main concepts and general parallelization strategies meta-heuristics, including their instantiation for neighbourhood-and population-based methods, recalls the main contributions to the field, and identifies a number of open questions and research challenges. The presentation is structured by a three-dimensional classification of design strategies for parallel meta-heuristics: the number of levels indicating whether decomposition is applied once only or recursively; the decomposition strategy reflecting the sources of parallelism in meta-heuristics, algorithm, search space, or mathematical structure; and the search strategy, given a particular level and decomposition approach, defined by the number of processes controlling the search, the communication and learning mechanism, and the diversity of the individual methods involved and their initial solutions. Six major classes of parallel meta-heuristics strategies are thus discussed: low-level decomposition of computing-intensive tasks with no modification to the original algorithm, explicit decomposition of the search space, independent multi-search, as well as synchronous, asynchronous, and knowledge-creating cooperative multi-search.

Keywords: Parallel computation, meta-heuristic search, parallelization strategies, algorithmic design taxonomy.

Acknowledgements. While working on the project, the author was Adjunct Professor, Department of Computer Science and Operations Research, Université de Montréal. The author gratefully acknowledges the financial support provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery grant program, and by the Fonds de recherche du Québec (FRQ) through their infrastructure grants.

---

* Corresponding author: teodorgabriel.crainic@cirrelt.net

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2024

1 Introduction

This chapter is concerned with the very broad domain of approximate solution methods, generally identified as *heuristics*, developed to address optimization problems of principally a combinatorial nature. More precisely, the chapter targets *meta-heuristics*, that is, heuristics involving at least two solution methods, one of which guides the global search for good solutions while avoiding the traps of local optima. Obviously, at least one of the solution methods involved is heuristic. The meta-heuristic term designates a very large array of neighbourhood (the term "trajectory" is also found in the literature) and population-based algorithmic designs, as witnessed by this handbook and several review/synthesis books, chapters and papers, e.g., [75, 1, 71, 96, 70, 69, 80, 16, 3, 120, 61, 17, 53, 92, 88, 62, 117, 81]. We include *matheuristics* [91] and *hyper-heuristics* [18] into the large meta-heuristic solution-method class, but do not address heuristics based entirely on exact solution methods, such as stopping early a Branch-and-Bound search or rounding the continuous values of the optimal solution to a linear relaxation of an integer program.

Two major goals characterize the development of meta-heuristics that take advantage of parallel computing. Common to all parallel computing development efforts, the objective is to solve larger problem instances, faster. That is, address larger problem instances than what is achievable by sequential methods, and do this in reasonable computing times. A second objective characterizes approximate solution methods and it concerns what some authors call the robustness of the method, that is, its capability to offer a consistently high level of performance over a wide variety of problem settings and instance characteristics. The *algorithmic design* of the parallel meta-heuristic search, also called *parallelization strategy*, correctly coded and implemented on an appropriate computing infrastructure, is core to achieving these objectives.

Indeed, the literature shows that, in appropriate settings, parallel meta-heuristics are more robust than sequential versions, while also generally requiring less extensive and expensive parameter-calibration efforts. The literature also emphasizes that a somewhat limited number of parallelization strategies may be identified, independently of their particular problem settings, initial sequential meta-heuristics, and implementation details, It also shows that most of these parallelization strategies yield new algorithms with behaviours different from those of their sequential counterparts.

The objective of this chapter therefore is to present an unified and up-to-date overview of the parallel meta-heuristics field. It synthesizes, classifies, and describes the main concepts and general strategies for the design of parallel meta-heuristics, including their instantiation for neighbourhood- and population-based methods, recalls the main contributions to the field, and identifies a number of open questions and research challenges. The chapter focuses on the *design* of parallel meta-heuristic search algorithms, rather than on their implementation using particular coding languages and computing architec-

tures. It does, however, identify new trends, challenges, and opportunities that some of the new computing-platform developments bring to the field.

The chapter updates and enhances the material presented previously, in particular Crainic [29, 30]. The interested reader may further consult a number of surveys, taxonomies, and syntheses, e.g., [7, 35, 47, 36, 2, 27, 34, 94, 28, 120, 38, 103, 46, 29, 56, 113, 33, 4].

The chapter is organized as follows. Section 2 *Meta-heuristics and Parallelism - A Taxonomy* introduces an enhanced classification of design strategies for parallel meta-heuristics, together with a discussion of the potential for parallel computing in meta-heuristics and a brief description of performance indicators for parallel meta-heuristics. Section 3 *Algorithm-based Parallelization Strategies* addresses so-called low-level approaches focusing on accelerating computing-intensive tasks without modifying the basic algorithmic design. Methods based on the explicit decomposition of the search space are treated in Section 4 *Explicit Search-Space Decomposition*, while strategies based on the simultaneous exploration of the search space by several independent meta-heuristics constitutes the topic of Section 5 *Implicit SSD - Independent Multi-search*. General cooperation principles are discussed in Section 6 *Cooperative Search* and are detailed in Sections 7 *Synchronous Cooperation* and 8 *Asynchronous Cooperation*. Section 9 *Knowledge Creation in Cooperative Search* is dedicated to strategies aimed at enhancing the performance of cooperation, by taking advantage of the exchanged data to create new information relevant to both the cooperating solvers and the global search. We conclude in Section 10 *Conclusions*.

2 Meta-heuristics and Parallelism - A Taxonomy

Parallel computing for meta-heuristics means decomposing the total task of finding the "best" solution to a given problem, that is the global computing work associated to the search for that solution, into smaller/simpler computing *tasks*. These tasks are then assigned to a number of available processors, to be performed concurrently (as much as possible, hopefully completely given sufficient computing resources). In the following, we refer to the *process* which performs a given task (one process per task), as well as to the *solver*, that is the solution method, associated to that process.

This section introduces the taxonomy used to define, describe, and analyze the parallelization strategies for meta-heuristics and the resulting main classes of parallel meta-heuristics proposed in the literature. The taxonomy unifies and enhances previous versions, in particular Crainic, Gendreau, and Toulouse [40], Crainic and Hail [34], and Crainic [29, 30]. This classification is sufficiently general to encompass the principal parallel meta-heuristic classes, while avoiding a level of detail incompatible with the scope

and dimension limits of the chapter. We conclude the section with a brief discussion of performance indicators for parallel meta-heuristics.

The taxonomy is structured along three main dimensions, compactly presented in Figure 1. The *Number of Levels* indicates whether decomposition is applied once only or recursively. Most parallel meta-heuristics found in the literature implement a *single-level decomposition*, noted *1L*, that is, the decomposition and search strategy are defined for the complete problem, yielding a number of tasks to be addressed concurrently (at each iteration of the parallel meta-heuristic). *Multiple levels* (*nL*) of decomposition are defined when one or several of those tasks may be further decomposed, the number of levels corresponding to the depth of the resulting task tree. Thus, for example, a *2L* strategy means some initially obtained (first level) tasks are further decomposed, while a a *3L* strategy implies a further decomposition of some of the 2nd-level tasks.

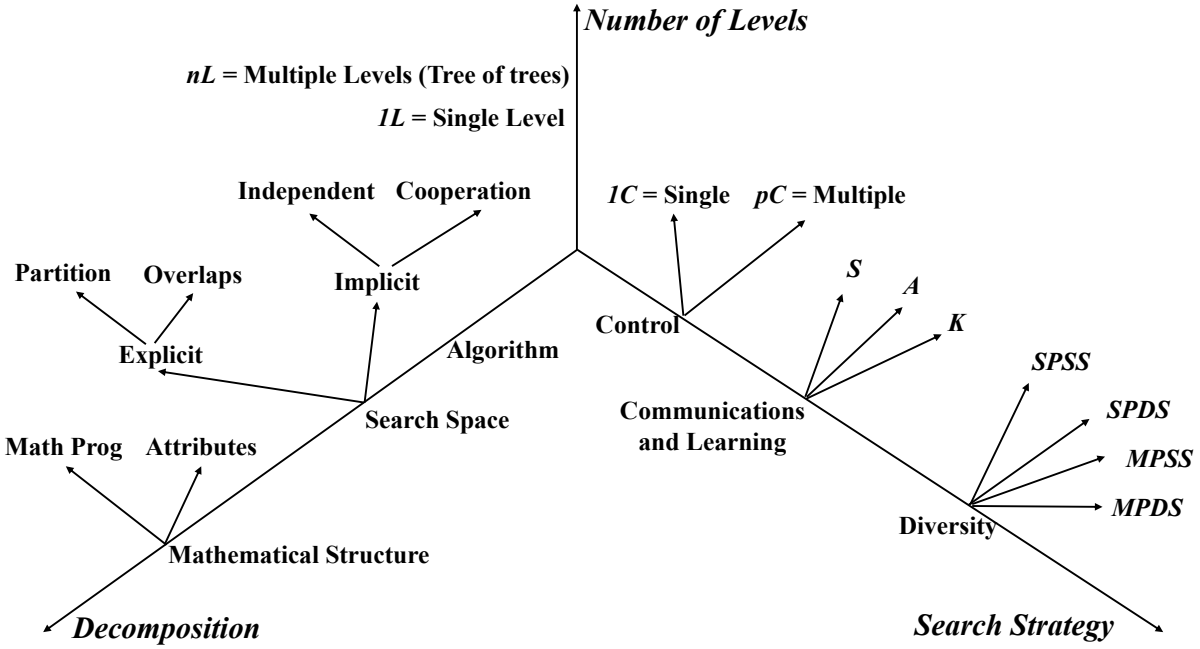

Figure 1: Parallel meta-heuristic taxonomy

The *Decomposition* dimension, detailed in Section 2.1 *Decomposition strategies: Sources of Parallelism*, reflects the sources of parallelism in meta-heuristics, in other words, what is decomposed in order to obtain the tasks to be addressed simultaneously. The decomposition may concern either the *Algorithm* (Figure 1) being parallelized, or the *Search Space* one associates to the formulation, or the *Mathematical Structure* of the problem and formulation.

The *Search Strategy* dimension focuses on the main parallelization strategies for meta-heuristics, given a particular level and decomposition approach. (Notice that previous versions of the taxonomy include this dimension only, the other two dimensions being ad-

dressed separately.) Section 2.2 *Search strategies* presents the three components defining a parallel meta-heuristic according to 1) *Control* of the global search with respect to the number of processes involved (one or multiple); 2) *Communication and Learning*, that is, how inter-process communications are handled **S**ynchronous or **A**synchronous) and whether those communications are used to gather information and build search-guidance **K**nowledge; and 3) *Diversity*, i.e., how the algorithms targeting the individual tasks are individually defined and instantiated: **S**ame or **M**ultiple initial **P**oints and **S**ame or **D**ifferent **S**earch algorithms.

The parallel meta-heuristic design and literature survey and synthesis of the next sections is based on this taxonomy. The characterization of the methods and contributions is captured through labels of the general form *L/Dec/C/C&L/Div*, where

***L*:** Number of levels

***Dec* =** Decomposition approach (see Section 2.1 *Decomposition strategies: Sources of Parallelism*)

- ***A*:** Algorithm
- ***SE*:** Search Space explicit decomposition
- ***SI*:** Search Space implicit decomposition - Independent multi-search
- ***SC*:** Search Space implicit decomposition - Cooperative multi-search
- ***MM*:** Mathematical structure decomposition - Mathematical programming
- ***MA*:** Mathematical structure decomposition - Attributes

Search Strategy (see Section 2.2 *Search strategies*):

- ***C* =** *1C*, *pC*: Type of control of the search strategy
- ***C&L* =** *S*, *A*, *K*: Communication and learning mechanism
- ***D* =** *SPSS*, *SPDS*, *MPSS*, *MPDS*: Solver and starting-point diversity

2.1 Decomposition strategies: Sources of parallelism

Parallel/distributed/concurrent computing means that several processes work simultaneously on several processors to address a given problem instance, aiming to identify the best possible solution for that instance. Parallelism thus follows from a decomposition of the total computational load and the distribution of the resulting tasks to available processors, which run in parallel, possibly exchanging information. According to how "small" or "large" the tasks are in terms of algorithmic work or search space, the parallelization is called *fine-* or *coarse-grained*, respectively. The decomposition may concern the algorithm, the search space, or the problem and formulation mathematical structure.

Also called "functional parallelism", *algorithm*-based decomposition focuses on computing-intensive parts of the algorithm. The goal is to separate the corresponding work into a number of tasks (processes or threads), working on the same data or on dedicated parts of the data, which are then allocated to different processors. The concurrent execution of the innermost loop iterations, e.g., evaluating neighbours, computing the fitness of individuals, or having ants forage concurrently, provides the main source of functional parallelism for meta-heuristics. In fact, this is often also the only source of readily available parallelism in meta-heuristics. Indeed, the execution of most other steps in a meta-heuristic algorithm depends on the status of the search, e.g., what has been performed so far and the values of the decision variables, which requires the computation of the previous steps to be completed. Even when this may be performed in parallel, the corresponding computational tasks must be synchronized to wait for the slowest one to complete, which generally yields significant delays, making such parallel computation non relevant. Section *Algorithm-based, Low-Level Parallelization Strategies* reviews the main developments and trends based on this decomposition approach.

Search space separation (*SSD*) constitutes a second major class of decomposition strategies, as parallelism for meta-heuristics may also be found in the domain of the problem formulation or the corresponding search space (for brevity reasons and without loss of generality, the term *search space* is used in this chapter).

The fundamental idea of the *explicit* SSD (the term "domain decomposition" is also found in the literature) is to create as many tasks as the number of resulting subspaces. Each task then addresses the original problem/formulation on its particular subspace. Notice that, theoretically, the parallelism present in the search space is as large as the space itself, provided a processor is assigned to each solution. Obviously, this is not practical. Hence a coarser decomposition is used in most cases of interest, and an exact or heuristic search method has to be assigned to each task.

As discussed in Section 4 *Explicit Search-Space Decomposition* reviews the main developments and trends based on this decomposition approach. The explicit SSD may proceed through a *partition* of the space, i.e., there is no intersection between any two subspaces created, or through a separation that allows *overlaps* between two or more subspaces. Both cases raise a number of issues with respect to the overall meta-heuristic search strategy, e.g., how to separate to create the tasks; how to create a complete solution out of the ones obtained on each subspace; how to allocate resources for an efficient exploration avoiding, for example, regions with poor-quality solutions. Trade offs must be examined and decided. Thus, for example, partitioning may simplify the treatment of some of these issues, e.g., reconstruction of complete solutions, due to its assurance that no data dependencies exist between the optimization processes of the various tasks, which may therefore be fully performed in parallel, yielding unique solutions for each subspace. Partitioning may hinter exploration, however, as solution transformations (e.g., moves and combinations of individuals) performed close to subspace boundaries overlook

possibilities involving solutions on the "other" side of the border. Allowing overlapping subspaces addresses this flaw, but requires a more careful construction of a complete solution out of partial solutions involving a certain number of the same variables with possibly different values.

Implicit SSD is performed through concurrent explorations of the search space by several solvers, which could implement the same solution method but start their exploration from different points of the search space. The final complete solution to the problem at hand is then selected when all solvers have completed their explorations. Implicit SSD belongs to the larger class of *multi-search* meta-heuristics (the terms "multiple walks" is also to be found in the literature), which also includes the algorithms produced through the attribute-based decomposition of the mathematical structure of the problem and model introduced below.

Two classes of implicit SSD strategies are defined. *Independent* search involves several solvers that do not exchange any information, except at the very end when the best solution is extracted. As discussed in Section 5 *Implicit SSD - The Independent Multi-search*, independent search strategies are very easy to implement and may display interesting performances. They are generally outperformed, however, by well-designed and implemented parallelization strategies based on *Cooperation*. Cooperative search is based on inter-process communication and information-exchange mechanisms. A rather broad spectrum of cooperation strategies may be defined, from the simple exchange of the current best solutions, to learning mechanisms based on exchanged data that yield new information to guide the global and individual searches. Cooperative strategies are at the core of the most successful developments in parallel meta-heuristics and are the object of the last three sections (preceding the *Conclusions* section).

Parallelism may also be obtained by decomposing along the mathematical structure of the problem and formulation. The tasks generated through such *mathematical structure*-based decomposition strategies address partial problems (called subproblems by a number of authors) obtained either through mathematical programming or attribute-based heuristic approaches. Mathematical-programming decomposition implicitly defines how a complete solution is built out of partial ones. In the latter case, some tasks work on the partial problems corresponding to the particular sets of attributes defined in the decomposition, while others combine the resulting partial solutions into complete solutions to the original problem. As described in Section 9 *Knowledge Creation in Cooperative Search*, attribute-based mathematical structure decomposition belongs to the class of multi-search strategies and is largely based on cooperation.

2.2 Search strategies

We build upon and enhance the classification of [34], generalizing that of [40], to describe the parallel search strategies for meta-heuristics and their characterizations in terms of how the global problem-solving parallel search is controlled, how information is exchanged among processes creating, eventually, new information, and the diversity of searches involved.

Control makes up the first characteristic (called "Search Control Cardinality" in previous versions). It specifies whether the global search is controlled by a *single* process or by *multiple* processes, which may collaborate or not. The two categories are identified as *1-control (*1C*)* and *p-control (*pC*)*, respectively. A process involved in the control of the global parallel search is named *controller* or *master*. The term *collegial* is used to designate a multiple-process control with collaborating masters, that is, when the controlling masters exchange information while the search proceeds and collectively steer the search path and determine its termination moment. (Recall that this chapter addresses the design of parallel meta-heuristics, not their implementation on particular computing-infrastructures, which always identifies a particular process in charge of distributing the initial work and collecting the appropriate information to determine that the search ends.)

The second search characteristic, *Communications and Learning*, addresses the issue of information exchanges among processes and the utilization of those exchanges to build new information, which may be used to control and guide the search (this characteristics was previously named "Search Control and Communications"). Two types of communications are generally identified in parallel computing: *synchronous* and *asynchronous*. In the former case, all concerned processes stop and engage in some form of communication and information exchange at moments exogenously determined (number of iterations, time intervals, specified algorithmic stages, etc.), either hard-coded or imposed by a control (master) process. In the latter case, each process is in charge of its own search and of establishing communications with other processes according to its own internal rules and status.

Learning in this context refers to the integration of mechanisms to 1) store all or part of the information exchanged, 2) derive/extract additional information from these exchanges and the stored data, and 3) use this enhanced knowledge to build a global view of the status of the parallel search and guiding information for the individual searches involved and, hence, for the global search.

Three categories are defined to reflect the communication type and the learning capabilities offered by a search strategy:

- *Synchronous* (S) communications with no learning ("Rigid Synchronization", RS,

previously)

- *Asynchronous* (A) communications with no learning ("Collegial", C, previously)
- *Knowledge* (K; replaces the previous "Knowledge Synchronous", KS, and "Knowledge Collegial", KC, categories)

More than one solution method or variant may be involved in a parallel meta-heuristic, and the methods may be meta-heuristics or exact. The third component of the Search Strategy dimension of the taxonomy describes the degree of *Diversity* of the global parallel search with respect to the nature of the methods involved and the corresponding starting solutions, the same or different in both cases (this dimension was identified as "Differentiation" in previous versions). Note that one characterizes two solvers as "different" even when based on the same methodology (e.g., Tabu Search, TS, or Genetic Algorithm, GA), provided they use different search strategies in terms of components (e.g., neighbourhoods or selection mechanism) or parameter values. The four classes are:

- *SPSS*: Same initial Point/Population, Same search Strategy
- *SPDS*: Same initial Point/Population, Different search Strategies;
- *MPSS*: Multiple initial Points/Populations, Same search Strategies;
- *MPDS*: Multiple initial Points/Populations, Different search Strategies

where "point" relates to neighbourhood-based single-solution methods, while "population" is used for population-based ones.

2.3 Performance measures

The traditional goal when designing parallel solution methods is to reduce the time required to "solve", exactly or heuristically, given problem instances or to address larger instances without increasing the computational effort. For exact solution methods running until the optimal solution is obtained, this translates into the well-known *speedup* performance measure, computed as the ratio between the wall-clock time required to solve the problem instance in parallel with p processors and the corresponding solution time of the best-known sequential algorithm. A somewhat less restrictive measure replaces the latter with the time of the parallel algorithm run on a single processor. See Barr and Hickman [10] and Schryen [114] detailed discussions of this issue, including additional performance measures.

Speedup measures are difficult to define, however, when the optimal solution is not guaranteed, including when the exact method is stopped before optimality is reached, which is obviously the case for meta-heuristics, in all senses of the term. Moreover, most strategies designing parallel meta-heuristics yield solutions that are different in value, composition, or both, from those of the sequential versions (when they exist). Hence, other than the acceleration of the search and the computational efficiency given the computing power committed to it, an equally important objective for parallel meta-heuristics is to what extend they outperform their sequential counterparts in terms of solution quality or, at least, support the claim of quality through a broader exploration of the search space. In other words, the parallel method should not require a higher overall computation effort than the sequential method or should justify the effort by higher quality solutions.

Search robustness is another characteristic increasingly expected of parallel heuristics. Here, robustness is defined with respect to a problem setting, in the sense of providing "equally" good solutions to a large and varied set of problem instances, without excessive calibration, neither during the initial development, nor when addressing new instances. As discussed by Crainic and Toulouse [35, 36], multi-search methods, particularly those based on cooperation, are noticeable in this respect. They display a behaviour quite different from those of the sequential methods involved, and offer enhanced performances compared to sequential methods and other parallelization strategies in terms of solution quality and method robustness. They are thus generally acknowledged as proper meta-heuristics in their own right [2].

3 Algorithm-based Parallelization Strategies

Functional-parallelism-based strategies, exploiting the potential for task decomposition within the inner-loop computations of meta-heuristics, aim to accelerate the search without modifying the algorithmic logic, the search space, and the behaviour of the sequential meta-heuristic. Hence the label *low level* often associated with such strategies, and the still prevalent utilization as the lowest level of hierarchical parallelization strategies, or when addressing problem settings requiring a significant part of the computing effort to be spent in inner-loop algorithmic components or in function evaluations. The utilization of "recent" computing elements increasingly ubiquitous within most computers, such as the *graphical processing units* (GPU), is reviving the interest in this class of strategies, as impressive reductions in computing times may be obtained.

Most low-level parallel strategies belong to the 1/A/1C/S/SPSS class, and are usually implemented according to the classical *master-worker* parallel programming model (the term "slave" is also abundantly found in the literature, particularly when the task concerns computing only, without any search). Typically, the exploration is initialized from

a single solution or population, and the "master" process executes the single-control sequential meta-heuristic, decomposing computation-intensive tasks into subtasks dispatched to "worker" processes. Worker subtasks are obtained from the same unique point, solution or population, and are essentially of the same type. Hence, the SPSS characterization of such strategies. Workers receive their tasks from the master, execute them in parallel, and return the results to the master which, once all the results are in, resumes the normal logic of the sequential meta-heuristic. The master has thus a complete view and control of the search status and execution; it decides the work allocation for all other processes and initiates communications. No communications take place among worker processes. Figure 2 illustrates this strategy (for a very limited number of processes, as in all figures in the chapter).

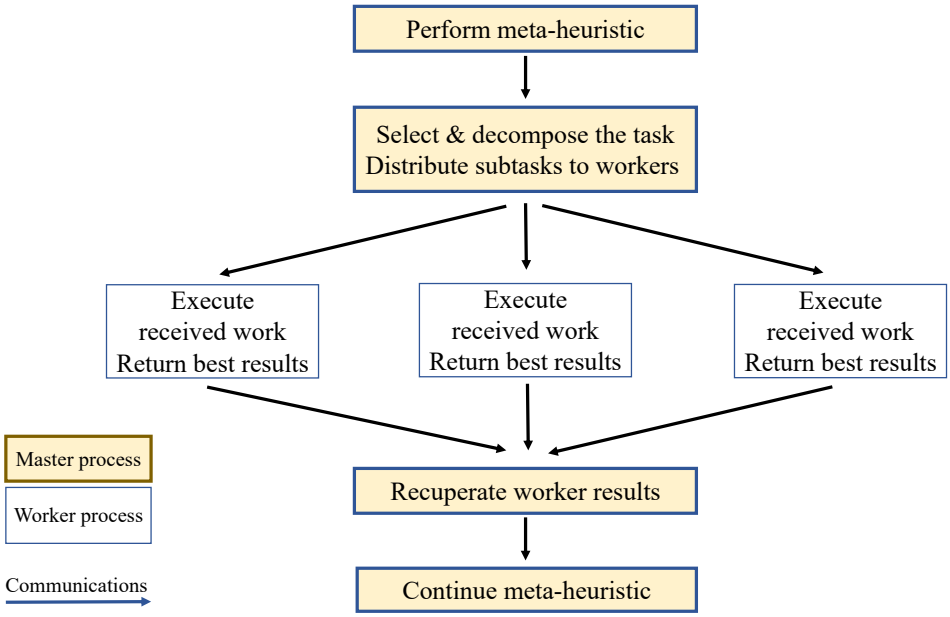

Figure 2: Low-level 1/A/1C/S/SPSS parallelization strategy

The neighbourhood-evaluation procedure of the local search heuristics, used alone or as component of neighbourhood- or population-based meta-heuristics (the latter implementing advanced "schooling" for offspring) is generally parallelized according to 1/A/1C/S/SPSS designs. The master groups the neighbours into tasks and sends them to workers. Each worker then executes the neighbour evaluation and, possibly, limited exploration (a few moves only out of each received neighbour or best-valued among them) procedure on its respective part of the neighbourhood, and sends back the best, or first improving, neighbour found. The master waits for all workers to terminate their computations, selects the best move, and proceeds with the search.

The appropriate decomposition granularity, that is, the size of the tasks, depends upon the particular problem and computer architecture, but is generally computationally sensitive to inter-processor communication times and work-load balancing. The most

approach most often encountered in the literature partitions the set of neighbours, uniformly (sequentially) distributing them into subsets of same cardinality, determined by the number of processes available. Such an uniform-partition approach is appropriate when the work associated to each neighbour evaluation is similar and the processors are homogeneous. This is not always the case, however, even when processors are of the same type. [49] illustrates this issue for the permutation-based local search neighbourhood (which performs very well within a sequential meta-heuristic) applied to the problem of scheduling dependent tasks on homogeneous processors. The dependency yields neighbourhoods of unequal sizes, requiring different evaluation efforts. Computational results then show that the uniform partition methods is not appropriate, the performance worsening with the amplitude of the variance in evaluation effort. A fixed coarse-grained non-uniform decomposition, based on the estimated computational effort of each subset, offers superior results, but it is not robust as it requires calibration each time the problem size or the number of processors changes. The best performing strategy, called *dynamic fine-grained* by the authors, defines each neighbour evaluation as a single task, the master dynamically dispatching these on a first-available, first-served basis to workers as they complete their tasks. The master still waits until all evaluations are completed, but the dynamic fine-grained strategy provides maximum flexibility and good load balancing.

Similar observations may be made regarding population-based meta-heuristics. In theory, all GA operators may be addressed through a 1/A/1C/S/SPSS design, and the degree of possible parallelism is equal to the population size. In practice, the computations associated to most operators are not sufficiently heavy to warrant parallelization, while overhead costs may significantly reduce the degree of parallelism and increase the granularity of the tasks. Consequently, the fitness evaluation is often the target of algorithm-based parallelism for genetic-evolutionary methods, usually implemented using the master-worker model. The master partitions the individuals among workers, which compute and return the fitness of each individual, as well as aggregate figures to facilitate the computation by the master of the average population fitness once it receives the reports of workers.

The algorithm-based parallelism for Ant-Colony Optimization (ACO) and, generally, swarm-based methods lies at the level of the individual ants. Ants share information indirectly through the pheromone matrix, which is updated once all solutions have been constructed. There are no modifications of the pheromone matrix during a construction cycle and, thus, each individual ant performs its solution-construction procedure without data dependencies on the progress of the other ants. Many parallel ant-colony methods proposed in the literature implement some form of 1C/S/SPSS strategy according to the master-worker model (e.g., [52] and references herein). The master builds tasks consisting of small colonies of one or a few ants, and distributes them to the available processors. Workers perform the construction heuristic and return their solution(s) to the master, which updates the pheromone matrix, returns it to the workers, and so on. To further

speed up computation, the pheromone update can be partially computed at the worker level, each worker computing the update associated to its solutions. This fine-grained version with central matrix update outperformed the sequential version of the algorithm in most cases. It is acknowledged, however, that it does not scale when implemented on "traditional" processors (i.e., exploiting the central processing units - CPUs), and that, similarly to other meta-heuristics, it is outperformed by more advanced multi-search methods.

Scatter Search (SS) and Path Relinking (PR) implement different population-based evolution strategies, where a restricted number of elite solutions are combined, the result being enhanced through a local search or a full-fledged meta-heuristic, usually neighbourhood-based. Consequently, the 1/A/1C/S/SPSS strategies discussed above apply straightforwardly, as in [58, 59, 57] for the p-median and the feature-selection problems. A different algorithm-based SS decomposition strategy may be obtained by running concurrently the combination and improvement operators on several subsets of solutions in the reference set. Here, the master generates tasks by extracting a number of solution subsets, which are sent to workers. Each worker then combines and improves its solutions, returning the results to the master for the global update of the reference set. Each subset sent to a worker may contain the exact or higher number of solutions required by the combination operator. In the former case, the worker performs an "iteration" of the SS algorithm [58, 59, 57]. In the latter case, several combination-improvement sequences could be executed and solutions could be returned to the master as they are found or all together at the end of all sequences. Load-balancing capabilities should be added to the master to avoid differences in work quantity and computing times between workers.

To conclude, algorithm-decomposition with 1-control parallel strategies are particularly attractive when neighbourhoods or populations are large, or when the neighbour or individual evaluation is costly. Computing time gains may then be obtained, as illustrated by many early contributions to the field (discussed in the surveys indicated in the Introduction), as well as in the parallelization of the Variable Neighbourhood Search (VNS) meta-heuristic proposed by [24]. The proposed parallel algorithm runs a single VNS, which executes in parallel the computations related to three steps: 1) identification of the initial solution: each process runs the local search out of a given starting solution; 2) the randomization ("shaking" in VNS vocabulary) of the solution provided by the local-search run: copies of the solution are distributed and each process performs a random modification; 3) the local search: the neighbourhood evaluation is distributed. Each time, the master selects the best alternative.

Impressive gains may be obtained by taking advantage of the current computing platforms integrating multi-core central processing units (CPUs - the "traditional" processor) and more recent types of computing units, graphical processing units (GPUs), in particular. Enhanced with data streaming capabilities, the latter provide hardware data

parallelism and the means for each processor to perform the same task on different (but rather small) parts of the distributed data [14, 15]. This hardware technology offers the possibility of extensive very low-level parallelization reminiscent of the work performed by the massively parallel computers of the late nineteen eighties. Neighbourhood evaluations, population fitness computation, and, even, the evolution of individuals in swarms may benefit from such a hardware-oriented parallelization, significant speedups having been observed (e.g., [95, 14, 15, 50, 23, 130, 121, 56, 108, 74]. A number of remarks are in order, however. First, the utilization of this technology is not straightforward, and work must be dedicated to its conceptual, technical and experimental aspects (see, e.g., [93]). Second, there is also the need to examine the sequential and parallel meta-heuristic designs to identify and valuate where this technology would bring the most benefits, besides those already identified. The work of [109] is a step on this research path. Finally, as discussed in the following sections, more advanced multi-search strategies outperform low-level strategies in most cases, in particular with respect to solution quality. Consequently, hierarchical settings combining multi-search strategies and 1/A/1C/S/SPSS evaluation procedures, all on CPU-based architectures, are generally used currently. More research is needed in this area to account for the massively parallel possibilities of new types of computing units.

4 Explicit Search-Space Decomposition

The basic idea of this class of strategies is intuitively simple and appealing: separate the search space into smaller subspaces, address the resulting subproblems by applying the sequential meta-heuristic on each, collect the respective partial solutions, and reconstruct an entire solution out of the partial ones. This apparently simple idea may take several forms, however, according to the type of separation performed, the permitted links among the resulting subproblems, the possible iterative modification of the separation, and the type of control of the parallel meta-heuristic.

As indicated in the taxonomy, the decomposition may yield *partition* or a *cover* of the complete search space. In the former case, the resulting subspaces are disjoint, while some overlaps are allowed among groups of subspaces in the latter. Thus, for example, the customer nodes, and the corresponding arc-design variables, of a Vehicle Routing Problem (VRP) may be partitioned into customer subsets (including the depot in each subset), while a cover would allow customers in a subset located "close by" another subset to belong to both. Note that covers may be defined implicitly by allowing the search within a given subspace to reach out to some part of one or several other subspaces through, e.g., neighbourhood moves or individual crossovers. The union of the subspaces makes up the complete space in all cases.

Strict partitioning restricts the solvers to their subspaces, resulting in part of the

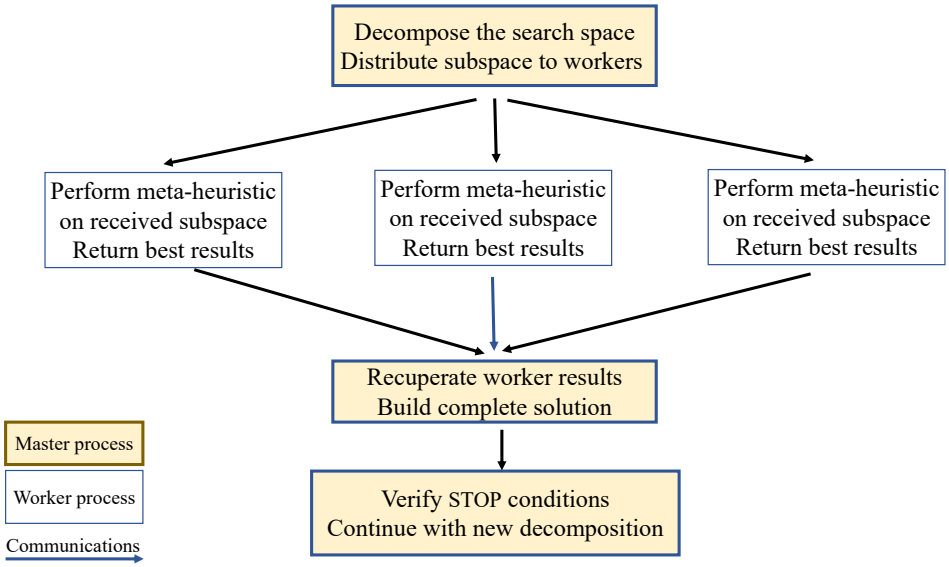

Figure 3: Explicit search-space single-control decomposition L/SE/1C/S/ strategy

search space being unreachable and the loss of exploration quality for the parallel meta-heuristic. Covers, through explicit or implicit overlapping, partially address this issue. Indeed, to guarantee that all potential solutions are reachable, one must make overlapping cover the entire search space, which would negate the benefits of decomposition. To avoid these drawbacks, one can change the separation and start again. This idea translates into a strategy encountered quite frequently, where the separation is modified periodically, and the search is restarted using the new decomposition. A complete-solution reconstruction feature is almost always part of the procedure. Note that, this approach also provides the opportunity to define non-exhaustive separations, i.e., where the union of the subspaces is smaller than the complete search space.

This strategy is naturally implemented using master-worker L/SE/1C/S schemes, with MPSS or MPDS search differentiation (Figure 3). The master determines the separation and sends partial subsets (or information to define them out of the initial space - this reduces the communication overhead) to workers, synchronizes them and collects their solutions, reconstructs complete solutions, modifies the separation, and determines when stopping conditions are met. Workers concurrently and independently perform the search on their assigned subsets. Most implementations addressed problem settings for which a large number of iterations can be performed in a relatively short time and restarting the method with a new decomposition does not require an unreasonable computational effort (see, e.g., [65] for real-time ambulance fleet management), a full-fledged meta-heuristic being generally used on each subspace.

Explicit space separation may also be performed in a multi-control decision-making framework with MPSS or MPDS search-differentiation (Figure 4). According to a L/S/pC/S

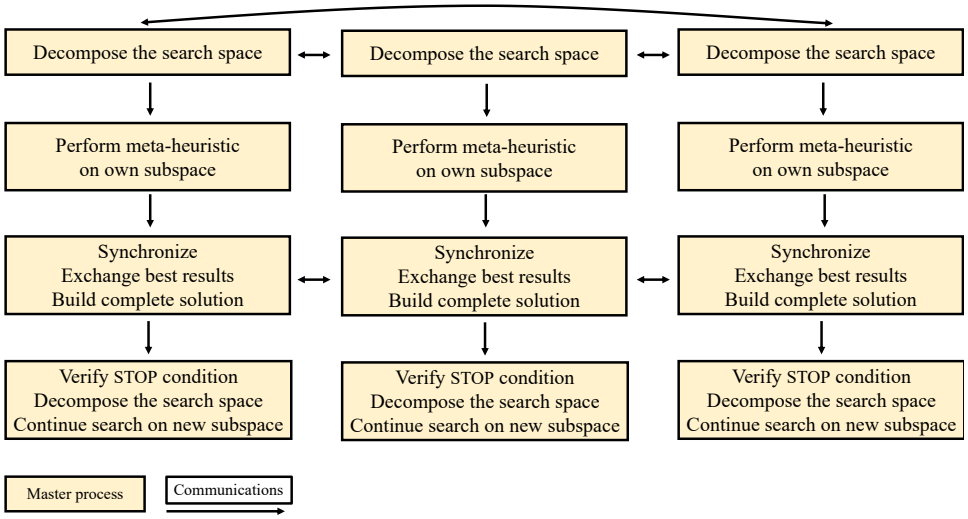

Figure 4: Explicit search-space multiple-control decomposition L/SE/pC/S strategy

strategy, the separation is collegially decided and modified following information-exchange phases (round-robin bilateral or many-to-many exchanges) activated at given synchronization points. Solvers may exchange their best solutions only (e.g., routes in a VRP), or add so-called context information that may be used when modifying the separation (e.g., unserviced customers and empty vehicles in a VRP [118]), respectively.

We conclude this section with two observations. First, we notice that explicit search-space decomposition corresponds to imposing restricted value ranges to all the decision variables of the formulation. In practice, this is rather performed by identifying a subset of variables, and corresponding constraints, eventually, and discarding or fixing the other variables and constraints, the goal being to obtain smaller, easier to address subproblems. However, it is not always possible, or even desirable, to discard. To illustrate, consider an explicit search-space decomposition according to subsets of demands for the VRP and the Multicommodity Capacitated Network Design problem (MCND). VRP demand is defined at customer nodes, and the decomposition separates those into subsets. One may then easily eliminate the VRP customers that do not belong to a given subspace, as well as the network links connecting them (the depot and depot-customer nodes must be included), and solve the resulting restricted VRP. MCND demand is defined by origin-destination pairs of nodes. Partitioning the set of OD demands is straightforward. But, eliminating the OD pairs not part of a given subset, as well as the links making up the paths connecting them, would result in unconnected networks and infeasible subproblems. Separation by variable fixing (and projection of the corresponding constraints) is thus preferable. It yields the desired search-space decomposition into smaller subproblems. It is also more general, offering increased flexibility, as variables may be fixed at values other than zero, and considering the complete vector of decision variables, provides the opportunity of a more thorough evaluation of solutions and possible changes.

The second observation is that explicit search-space decomposition strategies induce different search behaviour and solution quality compared to those of the sequential meta-heuristic. Their performance appears tributary to how well and efficiently one avoids overlooking important regions of the search space, while purposely advancing toward good complete solutions. The effort to achieve these goals if often too high compared to the final benefits, particularly with respect to the performance of an efficient cooperation-based meta-heuristic. Hence, using explicit search-space decomposition as stand-alone strategy does not present significant interest.

On the other hand, one observes a continuous increase in the complexity of the problem settings and the dimensions of the instancse one needs to address. In this context, combining search-space decomposition (through variable fixing, principally) and cooperative search appears very promising. The Integrative Cooperative Search [82] is a step in this direction (see Section 9 *Knowledge Creation in Cooperative Search*).

5 Implicit SSD - Independent Multi-search

We dedicate a section to the *independent multi-search* search-space decomposition strategy as it was among the first to be proposed in the literature, and is also the most simple and straightforward p-control parallelization strategy, generally offering an interesting performance.

Independent multi-search seeks to accelerate the exploration of the search space toward a better solution, compared to the classical multi-start sequential search. It proceeds by initiating simultaneous solvers,with or without different search strategies, from different initial points, and selecting at the end the best among the best solutions obtained by all searches (Figure 5).

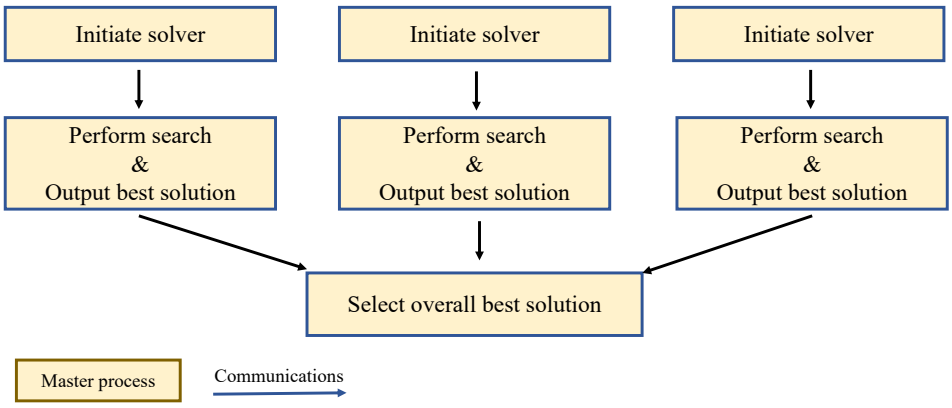

Figure 5: Implicit search-space multiple-control decomposition L/SI/pC/S strategy

Independent multi-search methods belong to the L/SI/pC/S class of the taxonomy, with any of the SPDS, MPSS, and MPDS diversity strategies. They do not change the behaviour of the corresponding multi-start heuristic. No attempt is made to take advantage of the multiple solvers running in parallel other than to identify the best overall solution at the final synchronization step. The efficiency of independent multi-search follows from the sheer quantity of computing power it allows one to apply to a given problem. The references identified in the Introduction describe numerous contributions of applying the independent multi-search strategy to a variety of combinatorial optimization problems.

Independent multi-search offers an easy access to parallel meta-heuristic computation, offering a tool when looking for a "good" solution without investment in methodological development or coding. Such methods are generally outperformed by cooperative strategies, however, through mechanisms enabling the independent solvers to share, during the search, the information their exploration generates. As explained in the following sections, this sharing and the eventual creation of new information out of the shared one, yields in most cases a collective output of superior solutions compared to independent and sequential search.

6 Cooperative Search

Cooperative (multiple or multi) search has emerged as one of the most successful classes of meta-heuristic methodologies to address hard optimization problems. While independent multi-search seeks to accelerate the multi-start sequential heuristic, search strategies based on cooperation go further and integrate *cooperation mechanisms* to share, *while the search is in progress*, the information obtained from this diversified exploration of the same problem instance. The global search behaviour of a cooperative meta-heuristic emerges from those interactions and sharing, including, eventually, the creation of new information out of the exchanged data. (The similarity between this behaviour and that of systems where decisions emerge from interactions among autonomous and equal "colleagues" has inspired the name"collegial control" associated to cooperative-search strategies in earlier versions of the taxonomy.) This behaviour is different compared to those of the solvers acting individually or in sequence (as in the multi-start case). It actually makes cooperative search a "new" meta-heuristic class in its own right, which provides in many cases better performances and solutions compared to sequential and parallel independent search [37].

Cooperative-search strategies are defined by the multiple solvers engaged in cooperation (pC control) and the *information-sharing cooperation mechanism* defining their interactions. The solvers define trajectories in the search space from possibly different initial points or populations, by using possibly different meta-heuristic or exact search

strategies. The goals of the cooperation mechanism are 1) to improve the performance of the solvers involved, and 2) to create as much as possible a global, "complete" image of the status of the cooperative search to enable guiding it, through participating solvers, toward a better performance, in terms of solution quality and computational efficiency, than the simple concatenation of results obtained by non-cooperating solvers.

A list of questions related to addressing this challenge that was initially proposed by Crainic, Gendreau, and Toulouse [123], and refined in later contributions, is still relevant today:

- What information to exchange?
- Between what processes to exchange?
- When to exchange?
- How to exchange?
- How to use/act on imported/received data?
- What knowledge to create and use out of exchanged data?

More than implementation details, answers to these questions constitute the core design parameters of a cooperative meta-heuristic. Thus, for example, a (not very good) cooperative mechanism has *all* solvers stop and *synchronize periodically*, exchange their *best local solutions*, and re-start their individual searches from the newly-identified *best overall solution*. Cooperation mechanisms proposed in the literature to answer these questions are described in the next sections, following a number of general observations.

Exchanged information must be *meaningful* and exchanges must be *timely*. "Good" solutions make up the most often exchanged type of information, usually taking the form of the overall best solution or the current-best solution of a solver being sent to the others. Broadcasting all the new best solutions a solver identifies is counter productive in most cases, however. This is particularly true when the solver performs a series of improving moves or generations, as solutions are generally "similar" (particularly for neighbourhood-based procedures), and the receiving solvers either have no chance to act on the in-coming information (unless special receiving mechanisms are embedded in all solvers) before receiving a new solution, or may embark on explorations similar to that of the sending solver. It was also observed that always broadcasting the overall best solution to all cooperating solvers is generally bad as it rapidly decreases the diversity of the search, increasing the amount of worthless computational work (many solvers will search in the same region) and bringing an early "convergence" to a not-so-good solution. Sending out the local optima after a series of improving moves, exchanging groups of solutions, and implementing random selection procedures for the solutions to

send out, the latter generally biased toward good or good-and-different solutions, are among the strategies aimed at addressing these issues.

Context information may also be shared profitably when embedded in the mechanisms used to guide the search. Context information refers to data collected by a solver during its own exploration, such as the statistical information relative to the presence of particular solution elements in improving solutions (e.g., the medium and long-term Tabu Search memories), the impact of particular moves on the search trajectory (e.g., the scores of moves of a Large Adaptive Neighbourhood Search), population diversity measures, individual resilience across generations, etc. A limited number of studies indicate the interest of context-information exchanges, but more research is needed on this topic.

Cooperating solvers may exchange information directly or indirectly. *Direct* exchanges of information occur either when the concerned solvers agree on a meeting point in time to share information, or when a solver broadcasts its information to one or several other solvers without prior mutual agreement. The latter case is to be avoided as it requires solvers to include capabilities to store received information without disturbing their own search trajectories until they are ready to consider it. Failure to implement such mechanisms generally results in bad performances, as observed for strategies combining uncontrolled broadcasting of information and immediate acceptance of received data.

Indirect exchanges of information are performed through independent data structures that become shared resources of data. Solvers may access them asynchronously and according to their own internal logic to post and retrieve information. Such data structures are called *blackboard* in the computer-science and artificial-intelligence vocabulary, while *memory*, *pool*, and *data warehouse* (*reference* and *elite set* are also sometimes used) are equivalent terms found in the parallel meta-heuristic literature. The term *memory* is used in this chapter.

Centralized-memory mechanisms have been used in most parallel meta-heuristic contributions. They receive, eventually process, and post information received from cooperating solvers, which, in turn, may retrieve this information independently. Distributed memory mechanisms may be contemplated, where a number of memories are inter-connected, each servicing a number of solvers. Such hierarchical structures, with several layers of solvers and memories, appear interesting when a large number of processors is involved, when computations are to take place on grids or loosely coupled distributed systems, and for integrative cooperation strategies. Issues related to data availability, redundancy, and integrity must then be addressed, as well as questions relative to the balancing of workloads and the volume of information exchanged. More research is needed on this topic.

Communications proceed according to an interaction topology represented by a *com-*

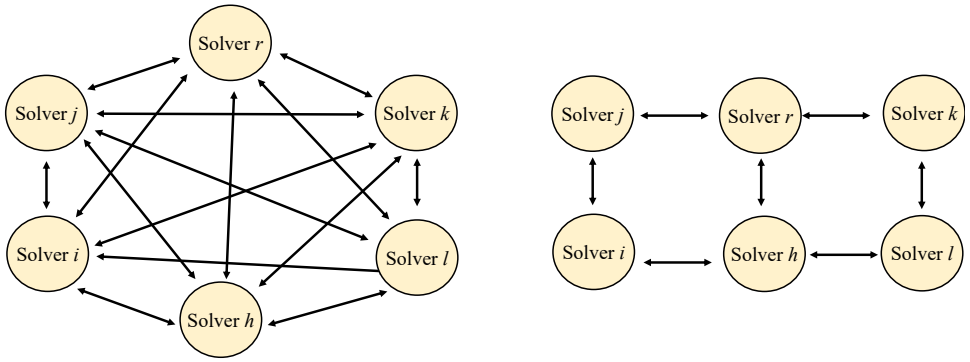

Figure 6: Interconnection communication graphs

munication graph specifying the processes that may engage in direct exchanges and, thus, directing the flow of information within the cooperative search. Each node of the graph represents a solver or a memory. Edges define pairs of solvers or a solver-memory pair, which may engage in direct communications. Figure 6 illustrates two communications graphs involving solvers only, a complete graph on the left, and a grid on the right, while communications pass through a central memory in the setting of Figure 7. The projection of this graph on the physical interconnection topology of the computer or computers executing the parallel program is generally part of the implementation design.

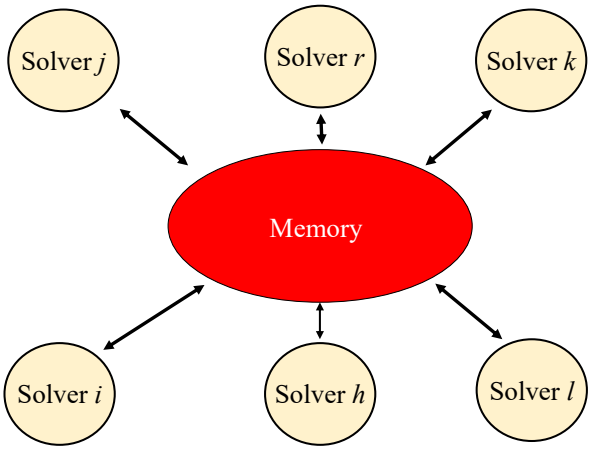

Figure 7: Indirect memory-based communication graph

When and how information is shared specifies the frequency of cooperation activities, who initiates them and when, and whether the concerned solvers must synchronize, i.e., each stopping its activities and waiting for all others to be ready, or not. These two cases

correspond to the *synchronous* and *asynchronous* characteristics of the Communications and Learning component of the taxonomy, and are discussed in the following sections. A general observation for both cases, however, is that exchanges should not be too frequent to avoid excessive communication overheads as well as premature "convergence" to local optima [125, 128, 124, 127].

Two observations to conclude this general discussion about cooperation. First, it is worth noticing that cooperation is somewhat biased toward intensifying the search in regions of the space that have already been explored and where interesting solutions have been identified. This is particularly true for "simple" cooperation mechanisms based on synchronization or on exchanging current-best solutions only. It is thus important to equip the cooperation strategies with diversification capabilities, such as, solver "welcoming" mechanisms to adapt incoming solutions to the local search environment and status (e.g., a crossover operator applied to the incoming and local solutions [90]), probabilistic or diversity-driven selection of exchanged solutions [131], or creation of new solutions and guidance information [82].

Second, the main principles of cooperative parallelization are the same for neighbourhood- and population-based meta-heuristics, even though denominations and implementation approaches may differ. We thus structure the presentation that follows based on these principles and general strategies, rather than by meta-heuristic class.

7 Synchronous Cooperation

Synchronous cooperation follows a p-control synchronous communication without knowledge creation (L/SC/pC/S) strategy, with any of the SPDS, MPSS or MPDS search differentiation approaches. We dedicate this section to recalling the main concepts of synchronous cooperation, some of which found their way into more advanced strategies, encouraging interested readers to consult the references indicated in the Introduction for details and references.

All the solvers involved in the cooperation stop their activities at particular moments and engage in an information-exchange phase, which must be completed before any solver can restart its exploration from that synchronization point. Synchronization moments may be determined by conditions imposed exogenously to all solvers (e.g., number of iterations from the last synchronization point), or detected by an a priori designated solver. Synchronization may use a complete communication graph or a more restricted, less densely connected topology (e.g., ring, torus, and grid graph; Figure 6). *Global* exchanges of information among all cooperating solvers take place in the former case, while information follows a *diffusion* process through *local* exchanges among neighbouring processes in the latter. In all cases, one aims to re-create a state of complete knowledge at

particular points in the global search and, thus, to hopefully guide it into a coordinated evolution toward the problem solution.

In a restricted view of the concept, most early 1/SC/pC/S cooperative search meta-heuristics based on global exchanges use a designated *master* process, which may or not include one of the participating solvers. The master manages the synchronization mechanism in a master-worker implementation. It initiates the global search starting the solvers, stops all solvers at synchronization points, gathers the information, updates the global data, verifies the termination criteria of the search and, either effectively terminates it or distributes the shared information (a good solution, generally, the overall best solution in many cases) and sends a signal to the solvers to continue the search. For coarse-grained island (each island corresponding to a sub-population) implementations of cooperating genetic methods, synchronization means the communication master initiates the *migration* operator to exchange among the islands the best or a small group of some of the best individuals in each. A similar approach is seen for ACO systems, dividing the colony into sub-colonies individually assigned to solvers, the master updating the pheromone matrix, and starting a new search phase, based on the received solver results [54]. A more sophisticated approach is proposed in [98] (and alluded to in [6]), where the master dynamically adjusts the search parameters of cooperating TS solvers according to the results each had obtained so far. The method performed well on the 0-1 Multi-dimensional Knapsack Problem, which is encouraging, as the idea of dynamic adjustment of the search parameters may be generalized to other problem settings and more sophisticated cooperation mechanisms.

A truer global pC/S cooperative scheme empowers solvers to initiate synchronization. Once it reaches a pre-determined status, a solver sends the stopping signal, broadcasts its data (current best solution or group of solutions, in most cases), followed by similar broadcasts performed by the other solvers. Once all information is shared, each solver performs its own import procedures on the received data and proceeds with its exploration of the search space until the next synchronization event. Most synchronous coarse-grained island parallelizations of GA-based evolutionary methods fall under this category, where migration operators are applied at regular intervals. Similarly, global synchronization for ACO applications where each colony evolves its own pheromone matrix, generally means that, after a fixed number of iterations, colonies exchange elite solutions that are used to update the pheromone matrix of the receiving colony. Expanding this idea to hierarchical designs, a 2/SC/pC/S strategy for genetic methods may have the fitness computation performed at the second level through a master-worker low-level parallelization [73]. Note that, the associated overhead due to the latter parallelization may became significant for larger numbers of processors.

Synchronization based on global exchanges of information generally incurs an excessive communication overhead and computing-time inefficiency as exchanges are initiated only when the slowest search thread is ready to start. The overhead may further increase

in hierarchical parallel contexts. A second main drawback of synchronous cooperation is that solvers relying heavily on the same information end up exploring the same search-space regions, which results in loss of diversity and favours premature "convergence" to local optima. Two approaches have been proposed to overcome this drawback.

The first approach is to avoid sharing local-best solutions only. The 1/SC/pC/S/MPDS iterated Tabu Search proposed for the VRP by [25] illustrates this approach, where solvers synchronize after a number of consecutive iterations without improvement within each individual search. Solvers then exchange a number of the good solutions obtained and, then, each individual solver builds a new starting solution by selecting routes probabilistically among those received and its own. Computational results showed this method to be flexible and efficient for several classes of routing settings with several depots, periodicity of demands, and time windows.

The second approach is based on *diffusion*, supported by sparse communication graphs (e.g., the grid in Figure 6). Then, direct communications at synchronization points are possible only with neighbouring solvers, i.e., with adjacent nodes in the graph. The quantity of information each solver processes and relies upon is thus significantly reduced. Information is still shared between non-adjacent solvers, but at the reduced diffusion speed of chains of local exchanges and data modifications by the intervening solvers. This idea was less explored compared to the global-exchange strategy, even though synchronous cooperative mechanisms based diffusion have a less negative impact on the diversity of the exploration and have yielded good results (e.g., [122, 97]).

To conclude, synchronous-cooperation strategies generally outperform the sequential versions and independent multi-search parallelization strategies. Yet, they also display a number of inherent drawbacks. Asynchronous information sharing addresses these challenges and, indeed, cooperation strategies based on asynchronous exchanges generally outperformed synchronous methods. We discuss these strategies in the next two sections.

8 Asynchronous Cooperation

Historically, multi-search independent and synchronous cooperative methods were the first to be developed. The focus has shifted to asynchronous cooperation, which may be considered as defining the "state-of-the-art" in parallel multi-search meta-heuristics.

A L/SC/pC cooperation strategy, with SPDS, MPSS, or MPDS search diversity, is asynchronous when solvers initiate exchange activities according to their own internal logic, without coordination with other solvers. Thus, e.g., a solver may make available its current best solution by posting it on a memory, or may ask for an external solution when it fails to improve the quality of its best solution for a certain number of iterations.

Asynchronous cooperative strategies follow either pC/A or pC/K communication-and-learning principles, the main difference being that the latter create new knowledge based on the information exchanged between solvers; pC/K strategies are addressed in the next section.

Asynchronous communications provide the means to build cooperation and information sharing among solvers without incurring the overheads associated with synchronization. They also bring adaptability to parallel cooperative meta-heuristics, which may more easily react and dynamically adapt to the exploration of the search space than independent or synchronous searches. These benefits come with potential issues one must care for, however. Two major ones are:

- The information related to the global search available to a solver performing its search may be less "complete" than in a synchronous environment; This implies that some solvers might explore more or less the same regions; The learning mechanisms discussed in the next section help addressing this issue.
- High data-exchange frequency may be damageable; It may disrupt the work of receiving solvers and induce an erratic search behaviour, similar to a random walk, particularly when the local search is altered significantly by the incoming data (e.g., the search is restarted from the received solution); Providing solver processes with mechanisms to "quarantine" incoming messages and data alleviate in part this issue, but at a generally high efficiency cost.

Hence the interest in applying information-sharing principles based on quality, meaningfulness, and parsimony [39, 40, 123]. With respect to the information shared, these principles mean:

- Exchange very good solutions or individual(s) only, rather than "all" locally improving ones; Many successful contributions in the literature involve sharing local optima only;
- *Diversify* the shared information. A diversifying strategy, for example, 1) collects the very good solutions sent by the solvers into a global *elite set*, and 2) returns to a requesting solver a solution selected randomly, but biased by quality, among the elite set. Such a strategy outperformed sending always the best available solution in the elite set [39].

When to initiate and perform cooperation activities, as well as how to use the incoming information are part of the parallel design with potentially important impact on performance. Most strategies proposed in the literature implement the same idea, to send and request information jointly. There is no need to do this, however, even though it can

decrease the amount of communications. It may thus be interesting for neighbourhood-based methods to make available right away their newly found local optima or improved overall solutions, and not wait for the algorithmic step when requesting and examining external information is appropriate. Similarly, population-based methods may migrate a number of individuals when a significant improvement is observed in the quality and diversity of their elite group of individuals.

Regarding the request for external information, it may be based on a a priory fixed number of iterations, but this approach should be restricted to meta-heuristics without search-diversification steps, e.g., Tabu Search based on continuous diversification. In most other cases, the principle of parsimonious communications implies selecting moments when the status of the search changes significantly, e.g., when the best solution or the elite subpopulation did not improve for a number of iterations. At such moments, solvers generally engage into some form of search *diversification* phase involving the choice of a different or modification of the current solution to initiate a new phase, e.g., the TS diversification based on long-term memories, the change of neighbourhood in VNS, and the complete or partial re-generation of population in population-based meta-heuristics. External information, which generally includes at least one good solution, may prove particularity interesting at that moment. How it is to be used depends on the particular logic of the receiving solver. It may be used, for example, to initiate a diversification phase, or to modify the search trajectory through a combination with a "local" solution, or to modify the solver behaviour in the long run through an insertion into an elite set. As already mentioned, one tries to avoid frequent imports followed by a replacement of the current solution or population, which generally results in a random search.

Direct and indirect exchange pC/A strategies may be used with any meta-heuristic. Historically, most GA-based asynchronous cooperative meta-heuristics relied on direct exchanges over complete [20] or sparse (e.g., ring) [89] communication graphs. These methods generally implement a coarse-grained island model, migration being triggered by conditions within individual island populations. The selected migrant individuals are then directed toward either adjacent islands when the graph is sparse or, when the graph is complete or dense, to all other populations or a dynamically-selected subset. The work of [129] illustrates the latter case, where migration is initiated by an island that identifies a new best solution, which it sends to all other islands. To enforce diversity, the migrant is accepted only when different from the local population and better than the worst individual in that population. We also mention the work of [76] who introduced genetic solvers with different strategies, which was a novelty in the GA-island field (previously, all island populations were evolved by the same algorithm), and observed significant improvements compared to more traditional island-based pC/A models. The parallelization of ACO methods may use the same approach, where partitions of the initial colony play the role of islands. The contribution of [87] is interesting in this context for novel way of selecting the receiving subcolony (island). Here, a solver initiates an exchange when the evolution of its colony becomes stagnant (no longer improving) by

selecting an exchange partner probabilistically based on the relative distance (the most different best solution) and fitness (of the best solution); it then requests the current best solution from the selected partner, and, upon reception, updates its pheromone matrix and continues the search. An interesting extension of the island-cooperation idea to 2-level strategy defines meta-heuristic-specific "archipelagos" of islands, with particular Communication-and-Learning strategies at each level [5]. The authors propose GA and ACO archipelagos with asynchronous cooperation between them, while synchronous exchanges are enforced among the islands of each.

Relatively little attention has been dedicated to direct exchanges for asynchronous cooperation among trajectory-based meta-heuristics. We mention the work of [116] reporting encouraging results for VNS-based 1/SC/pC/A strategies over uni and bidirectional ring topologies. More research on this topic is needed.

Historically, the sharing of information in most asynchronous cooperative search strategies outside the genetic-evolutionary community is therefore based on some form of indirect communications through a centralized device, the *central memory* illustrated in Figure 7 [39, 40, 27]. A solver involved in such a cooperation deposits good solutions, local optima generally, into the central memory, from where, when needed, it also retrieves information sent by other cooperating solvers. Classical retrieval mechanisms are based on random selection, which may be uniform or biased to favour solutions with high rankings based on solution value and diversity. The central memory accepts incoming solutions for as long as it is not full, acceptance becoming conditional to the relative interest of the incoming solution compared to the "worst" solution in the memory, otherwise. Diversity criteria are increasingly considered, a slightly worse solution being preferred if it increases the diversity of solutions in the central memory. Population culling may also be performed (deleting, e.g., the worst half of the solutions in memory).

Central-memory-based cooperative search strategies are described in the literature for most meta-heuristic classes. To the best of our knowledge, [39] was the first to propose a central-memory approach for asynchronous Tabu Search in their comparative study for a multi-commodity location problem with balancing requirements. In their proposed parallelization, individual TSs send to the memory their local-best solutions when improved and import a solution selected probabilistically biased by rank before engaging in a diversification phase. This method outperformed, in terms of solution quality, the sequential version as well as several synchronous and broadcast-based asynchronous cooperative strategies. The same approach was applied to the fixed cost, capacitated, multicommodity network design problem with similar results [32].

Asynchronous cooperation strategies with some form of central memory were proposed for a variety of problem settings, e.g., cutting [12], container loading [13], labour-constrained scheduling [22], and VRP with time windows (VRPTW) [84]. Another set of studies focused on asynchronous cooperation with indirect communications ap-

plied to particular classes of meta-heuristics, including Simulated Annealing (SA), e.g., [86, 111, 9]; VNS, e.g., [41, 105], the latter proposing a self-adapting mechanism for the main search parameters based on recent performance, and solution selection out of the ten best present in memory; GRASP with cooperation based on applying Path Relinking to solutions from memory [107]; and TS with memory hosting a reference set and long-term global memories built on short-term local memories sent by solvers [77].

Notice that cooperating solvers need not belong to the same meta-heuristic class. The next section presents several examples where different meta-heuristics collaborate within pC/K strategies. In the classical pC/A case, we find contributions following the same broad strategy described above when calling sequentially on meta-heuristics belonging to different types. The two-phase approach of [60] for the VRP with Time Window (VRPTW) is a typical example of such a method, where each solver first applies an evolution strategy to reduce the number of vehicles, followed by a TS to minimize the total distance travelled. A somewhat different two-phase pC/A parallel strategy is proposed in [11] for the Steiner problem, where each phase, using Reactive TS and PR respectively, implements the pC/A asynchronous central memory strategy, all processes switching from the first to the second phase simultaneously.

Multi-level cooperative search proposes a different pC asynchronous cooperative strategy based on controlled diffusion of information [126]. Solvers are arrayed in a linear, conceptually vertical, communication graph and a local memory is associated with each. Each solver works on the original problem but at a different level of aggregation or "coarsening" of the problem data (e.g., selected network arcs being collapsed into nodes that inherit some of the arc attributes). The solver addressing the complete original problem works at the lowest aggregation level, identified in Figure 8 as Level 0. Level L identifies the highest coarsening defined for the problem. Solvers communicate exclusively with the two solvers directly above and below, that is, at higher and lower aggregation levels respectively (obviously, workers at levels 0 and L exchange with the next higher and lower level, respectively).

Coarsening reduces the dimensions of the problem instance at hand, which becomes easier to address. One may thus identify very good solutions faster. Moreover, one expects that a very good solution to an aggregated version of the problem will yield at least a good solution when projected on a more disaggregated problem setting. This projection process is called *refinement*. The idea of a coarsen-refine method therefore is to successively coarsen the instance until the desired resolution level and then to successively refine that solution until feasible to the original problem setting. The multi-level cooperation builds on this idea through asynchronous up and down communications among adjacent solvers, each taking advantage of the structure and quality of the solution received from the next lower or higher solver. The communications are regulated by the local memories that receives the information coming from the immediate neighbours and makes it accessible to the local solver at moments dynamically determined according to

its internal logic.

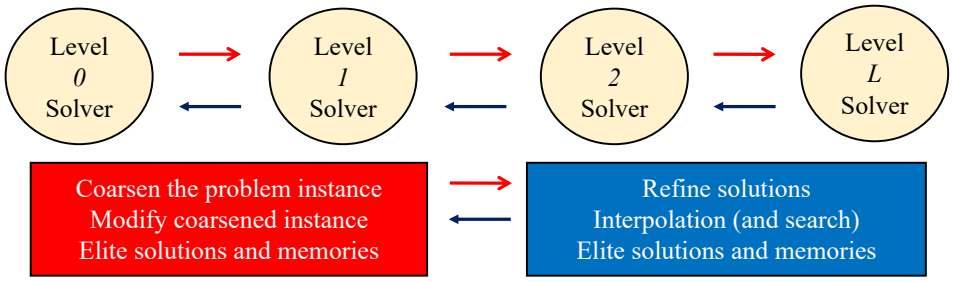

Figure 8: Multi-level cooperation

In the original implementation, solvers were exchanging improved solutions, incoming solutions not being transmitted further until modified locally for a number of iterations to enforce the controlled diffusion of information. Excellent results were obtained for various problem settings including graph and hypergraph partitioning [101, 102], network design [43], feature selection in biomedical data [99], and covering design [48]. It is noteworthy that one can implement multi-level cooperative search through a central memory by adequately defining the communication protocols. Although not yet fully defined and tested, this idea is interesting as it opens opportunities for richer exchange mechanisms combining controlled diffusion and general availability of global information.

Central-memory asynchronous cooperation strategies are generally offering very good results, yielding high-quality solutions. They are also computationally efficient as no overhead is incurred for synchronization. No broadcasting is taking place and there is no need for complex mechanisms to select the solvers that will receive or send information and to control the cooperation. It has also proved efficient in handling the issue of premature "convergence" in cooperative search, by diversifying the information received by the solvers through probabilistic selection from the memory and by a somewhat large and diverse population of solutions in the central memory; solvers may thus import different solutions even when their cooperation activities are taking place within a short time span. The central memory is thus an efficient algorithmic device that allows for a strict asynchronous mode of exchange, with no predetermined connection pattern, where no solver is interrupted by another for communication purposes, but where any solver may access at all times the data previously sent out by the other solvers.

The performance of central-memory cooperation and the availability of the exchanged information kept in the memory has brought up the question of whether one could design more advanced cooperation mechanisms taking advantage of the information exchanged among cooperating solvers. The pC/A strategies described in the next section are the result of this area of research.

9 Knowledge Creation in Cooperative Search

Cooperation, particularly in the central-memory asynchronous form, offers many possibilities for algorithm development. Particularly noteworthy are the flexibility in terms of the different meta-heuristic and exact methods that can be combined, and the population of elite solutions being hosted in the central memory and continuously enhanced by the cooperating solvers. One can thus select cooperating methods that complement each other, some of which heuristically construct new solutions, execute neighbourhood-based improving meta-heuristics, evolve populations of solutions, or perform post-optimization procedures on solutions in memory.

The study of [31] illustrates the interest of these ideas. The authors combine a GA solver and several solvers executing the pC/A Tabu Search for multicommodity location-allocation with balancing requirements of [39]. The TSs aggressively explore the search space, building the elite solution set in the central memory, while the GA contribute toward increasing the diversity, and hopefully the quality, of the solutions in the central memory, to be imported by the cooperating TSs. The GA launches with a population composed of the initial set of elite solutions generated by the TS solvers. Asynchronous migration subsequently transfers the best solution of the genetic pool to the central memory, and solutions in the central memory toward the genetic population. This strategy did perform well, especially on larger instances. It also yielded the interesting observations that 1) including different types of solvers in the cooperation is beneficial, and 2) while the best overall solution might have never been found by the genetic solver, its inclusion allowed the TS solvers to find better solutions, more diversity among solutions in memory translating into a more effective diversification of the global search.

Following the studies on memory and learning preformed for Tabu Search and synthesized in [70], several contributions in the parallel meta-heuristic literature, including the pioneering [110] and [31], focused on the utilization of the information exchanged among cooperating solvers to construct and continuously enhance an understanding of the history and status of the global search. The analysis of the elite population solvers build in memory, together with associated context data, becomes a *learning mechanism* providing information relative to, e.g., regions of the search space already explored and the relative quality of solutions identified in those regions, the performance of the cooperating solvers given the information received from the central memory, the behaviour of critical decision variables (e.g., arc or node selection in network design) relative to good solutions, etc. This information may then be used to create new *knowledge* to guide the search. Such knowledge may include new and diverse solutions or components, "ideal" target solutions, attribute-based patterns to bias probabilistic selections of solver moves, etc.

Cooperative strategies including mechanisms to create new information belong to the *p-control, knowledge communications-and-learning* pC/K class of the taxonomy. Most

contributions to this field have cooperating solvers work on the complete problem, a few being dedicated to the learning mechanism. The bulk of the section is dedicated to these strategies. We conclude the section with a discussion on developments targeting the *attribute-based decomposition of the mathematical structure* of the problem at hand, where solvers work on particular parts of the problem or on integrating the resulting partial solutions into complete ones.

Historically, two main classes of pC/K cooperative mechanisms are found in the literature, both based on the idea of exploiting a set of elite solutions exchanged by cooperating solvers working on the complete problem, but differing in the information kept in memory. *Adaptive-memory* methods [110, 119] store and score partial elements of good solutions and combine them to create new complete solutions that are then improved by the cooperating solvers. *Central-memory* methods exchange complete elite solutions among neighbourhood and population-based meta-heuristics and use them to create new solutions and knowledge to guide the cooperating solvers [27, 36, 39]. Notice that the latter method generalizes the former and, thus, the two are becoming increasingly unified.

The adaptive-memory terminology for parallel meta-heuristics was coined in a paper [110] proposing TS-based heuristics for routing problems (see [8] for a thorough application of the concept to the VRPTW). More comprehensive presentations of adaptive-memory concepts, applications, and challenges may be found in [68] and [119]. The main idea is to keep in memory the individual components (vehicle routes in VRP) making up the elite solutions found by the cooperating solvers, together with memories counting for each component the frequency of inclusion in the best solutions encountered so far, as well as its score and rank among the population in memory, computed in particular from the objective value of its respective solutions. Solvers construct solutions out of probabilistically selected (biased by rank) solution components in memory, enhance them (TS in the initial contribution), and deposit their best solutions in the adaptive memory. The probabilistic selection yields, in almost all cases, a new solution made up of components from different elite solutions, thus inducing a diversification effect.

A number of early developments provided insights into algorithmic design. Applying the concept to the VRP, [115] proposes a set-covering heuristic to select from the memory the components to generate the new initial solution of a cooperating solver. At about the same time, [64] explores 2-level hierarchical strategies in the context of real-time vehicle routing and dispatching. A cooperating adaptive-memory SC/pC/K/MPSS strategy is applied at the first level, where each TS solver implements a SE/1C/S route-decomposition with the help of several workers on the second level.

Generalizing the pC/A and adaptive-memory strategies, pC/K central-memory mechanisms keep full solutions, as well as attributes and context information sent by the solvers involved in cooperation (in later versions, partial solutions may also be kept). The central memory also keeps newly created information out of the exchanged data.

Classical statistics-based memories recording the performance of individual solutions, solution components, and solvers provide part of this new knowledge. More advanced strategies include information extraction and creation [85], machine learning mechanisms [19], and new solution-creation procedures (population-based, generally). Search guidance mechanisms based on this knowledge may thus be gradually built.

Solvers indirectly exchange complete elite solutions and context information though the central memory. Solvers may perform constructive, improving and post-optimization heuristics [84, 85, 82], neighbourhood-based methods like TS and GRASP [51, 78, 79, 82, 21], population-based methods like GA [84, 85, 51, 82], SS [104], and PR [42], as well as exact solution methods [72] on possibly restricted versions of the problem. The particular solvers to include in cooperation depend on the application. They should be efficient for the problem at hand, of course. Additionally, they should also aim to cover different regions of the search space in such a way that they contribute not only to the quality but also to the diversity of the elite population being built in the central memory.

Central-memory mechanisms thus perform two main tasks: data-warehousing and communications with solvers, on the one hand, information-creation and search-guiding, on the other hand. To distinguish between the two, we single out the latter as the *Search Coordinator* (*SC*). The simplest SC mechanism was used in the pC/A strategies of the previous section, where solutions in memory were ordered and rank-biased randomly extracted to answer solver requests. More sophisticated mechanisms are briefly described in the following.

The cooperative meta-heuristic proposed by [84] for the VRPTW uses a simple pC/K mechanism, involving four solvers, two GAs, with order and edge recombination crossovers, respectively, and two acknowledged TSs, Unified Tabu Search [26] and TABUROUTE [63]. The cooperating solvers share their improved best solutions, and request solutions as needed, i.e., the GAs for crossover operations, the Unified Tabu at regular intervals, and TABUROUTE at diversification time. The central-memory SC performs post-optimization procedures to reduce the number of vehicles and the total travelled distance) on the received solutions before making them available for sharing. This algorithm, without any calibration or tailoring, proved to be competitive with the best meta-heuristics of its day in linear speedups.

A general learning and guidance mechanism based on an atomic network-optimization elements, the arc in particular, is proposed in [85]. The SC is thus independent of particular problem characteristics (e.g., routes in the original VRPTW application), and may thus be broadly applied to network-based problem settings. Starting from the classical memory concepts pioneered for TS [66, 67, 70], the authors combined two ideas: first, that an arc appearing often in good solutions and less frequently in bad solutions may be worthy of consideration for inclusion in a tentative solution, and vice versa; second, that this worthiness increases when the behaviour appear stable in time. The authors

thus consider the evolution of the frequency of inclusion of arcs in solutions of different quality, that is, in the elite (e.g., the 10% best), average (between the 10% and 90% best), and worst (the last 10%) groups of solutions in the central memory. *Patterns* of arcs are then defined representing subsets of arcs (not necessarily adjacent) with similar frequencies of inclusion in particular population groups. Guidance is obtained by transmitting arc patterns to individual solvers, indicating whether the arcs in the pattern should be "fixed" or "prohibited" to intensify or diversify the search, respectively. Solvers account for these instructions by using the patterns to bias the selection of arcs for move or reproduction operations. A four-phase fixed schedule (two phases of diversification at the beginning to broaden the search, followed by two intensification phases to focus the search around promising regions) was used with excellent results in terms of solution quality and computing efficiency compared to the best-performing methods of the day (see [83] for a dynamic version of this mechanism).

A different SC mechanism for a pC/K meta-heuristic with TS solvers was proposed by [79] for the VRP. Data sharing is relatively simple; solvers periodically (after a number of iterations or when the solution did not improve for a number of iterations) sent best solutions to the central memory, and received a solution back from it, the search being resumed from the received solution. The SC mechanism's objective is to identify and extract information from the solutions in memory to guide solvers toward intensification and diversification phases. This is obtained by dynamically (on reception) clustering solutions according to the number of edges in common. Thus, solutions in a given cluster share a certain number of edges, this set of edges and solutions being assumed to represent a region of the search space. Search history indicators are associated with clusters giving the number of solutions in the cluster and the quality of the solutions. This information is used to infer how thoroughly the corresponding region has been explored and how promising it might still be. Clusters are sorted according to the average solution value of their feasible solutions, and the cluster with the lowest value, that is, with the largest number of very good solutions, is selected for intensification, while the solution with the lowest number of good solutions is selected for diversification (recall that the authors were addressing a minimization optimization problem). A solution is then selected in the corresponding cluster and it is sent to the requesting solver. Excellent results were obtained in terms of solution quality and computation effort (an almost linear speedup was observed with up to 240 processors) compared to the state-of-the-art methods of the day.

The study of [19] focuses on the issue of flexibility and adaptability of cooperative parallel meta-heuristics. The proposed cooperation involves a number of solvers of different types exchanging through a central memory. An additional process is connected to the central memory and is in charge of the SC mechanism. collecting information from the other solvers and sending guidance instructions to modify their behaviour. The SC is based on a set of fuzzy rules initially instantiated through a supervised knowledge extraction process. The rules are meant to decide when to reinitialize a solver either

with the current best solution or a new set of parameter values. The former aims at changing the behaviour of the currently "worst" performing solver, bringing it within a portion of the search space closer to the overall best solution. The latter also focuses on the "worst" performing solver, changing the parameter values which did not change for some time. Encouraging results were obtained on $\{0, 1\}$-knapsack problem, which indicates that more research is needed on this topic. The flexibility and adaptability issue is also at the core of the work of [100]. Set within the hyperheuristic context [18, 17], the cooperation involves three solvers based on, respectively, a GA, a SA, and an AOC. The solvers stop after a certain number of solver-specific measures and store their best solutions in the central memory. The SC mechanisms ranks the solvers based on performance, e.g., computing time and solution value. The idling solver with the highest ranking then receives a new intital solution from memory and restarts its search.

The pC/K/MPDS method proposed in [72] for the VRP illustrates how specialized solvers may address different issues in a cooperative meta-heuristic, including the generation of new knowledge. Two types of solvers were defined. The so-called heuristic solvers improve solutions received from the SC associated with the central memory. On completing the task, the solvers return the a given number of the best solutions found and the corresponding routes (a post-optimization procedure is first run on each route). Simultaneously, exact solvers aim at identifying new solutions by solving series of set covering problems, starting from a limited set of routes. Each time a set covering problem is solved, the solution is returned to the central memory and a pre-defined number of the current best solutions is retrieved for the next run. Set-covering solvers had also access to the ordered list of best routes in memory and they selected within to complete their problems. The number of routes selected to set up a set covering problem is dynamically modified during the search to control the corresponding computational effort. The method performed very well, both in terms of solution quality and computational effort (an almost-linear speedup was observed).

The inclusion of specialized solvers in cooperation is also part of the algorithmic design proposed in [112] The cooperative algorithm involves meta-heuristic solvers to construct and improve solutions, as well as a somewhat exact solver generating new variables (column generation of partial solutions, heuristically extended to complete ones through) to further diversify the search. The SC mechanism guides the cooperative meta-heuristic by sending solutions to the appropriate solvers to intensify or diversify the search.

The parallel GRASP algorithm proposed in [21] makes use of different information-sharing strategies at different stages of the parallel meta-heuristic. Each solver runs the same GRASP meta-heuristic, sharing not only its best solutions, but also results of intermediate computations, which reduces the computing time spent performing redundant operations over diverse solvers. Thus, the data updates and corresponding evaluations performed by a solver while building its initial solution are immediately stored in the

common central memory for access by all other solvers. Solvers also make immediately available their improved best solutions found during their local search phases. Finally, the parallel meta-heuristic proceeds in two phases. The first runs the solvers without external limits for a given duration. Solvers synchronize at the end of the first phase, their best solutions and context information are evaluated and aggregated, yielding a restricted set of data on which solvers will work during the second phase. The parallel meta-heuristic proved very efficient on three different problem settings.

We complete this section by reviewing developments targeting multi-attribute, "rich", problem settings displaying a large number of attributes characterizing their feasibility and optimality structures. Traditionally in the literature, such problems are simplified, or sequentially solved through a series of particular cases, where part of the overall problem is fixed or ignored, or both. The main idea of this generation of pC/K meta-heuristics is to decompose the mathematical structure of formulation along sets of decision variables, called *decision-set attribute decomposition* in [82]. The goal of this decomposition is to obtain simpler but meaningful problem settings, in the sense that efficient solvers can be "easily" obtained for the partial problems either by opportunistically using existing high-performing methods or by developing new ones. Thus, an opportunistic rule may decompose the multi-depot periodic VRP (MDPVRP) along the depot and period decision sets to create two partial problems, a periodic VRP (PVRP) and a multi-depot VRP (MDVRP), high-quality solvers being available in the literature for both problems. The central-memory asynchronous cooperative search framework then brings together these partial problems and their associated solvers, together with integration mechanisms, reconstructing complete solutions, and search-guidance mechanisms.

According to our best knowledge, [42, 51] were the first to propose such a methodology in the context of designing wireless networks, where seven attributes were considered simultaneously. The proposed 1/MA/pC/K/MPDS meta-heuristic has TS solvers working on limited subsets of attributes, the others being fixed, and a GA combining the partial solutions generated by the TS procedures into complete solutions to the initial problem.

The general method, called *Integrative Cooperative Search ICS*), was introduced in [82] (see [44, 45] for initial developments) and illustrated through an application to the MDPVRP . As illustrated in Figure 9, the main components of ICS, which need to be instantiated for each application, are the

- Decomposition rule;
- *Partial Solver Groups* (*PSG*) addressing the partial problems resulting from the decomposition;
- *Integrator Group* (*IG*) selecting partial solutions from PSGs, combining them, and sending the resulting complete solutions to the

- *Complete Solver Group* (*CSG*), which provides the central memory functionalities of ICS and, possibly, enhances the complete solutions.

Notice that, in order to facilitate the cooperation, a unique solution representation, obtained by fixing rather than eliminating variables when defining partial problems, is used throughout ICS. Also notice that memories associated to the PSGs and the CSG need not be distributed on different computing units (the display in Figure 9 is for illustration only). As for all parallelization strategies discussed in this chapter, the implementation is specific to the available computing infrastructure.

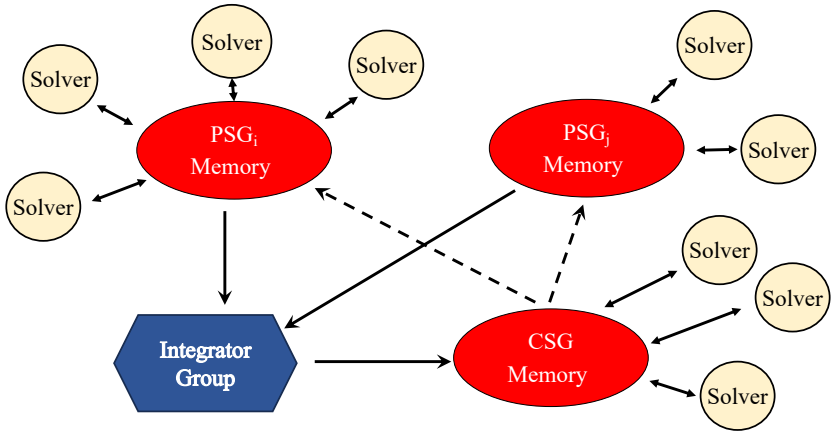

Figure 9: Integrative cooperative search

A Partial Solver Group may contain one or several solvers targeting the particular partial problem assigned to the group. When several solvers are involved, the PSG is organized according to a pC/K strategy, with its own central memory keeping partial elite solutions and associated context information. The PSG's SC manages the central memory and the exchanges with the other components of the ICS, the CSG in particular. Two PSGs were defined in [82], one for the PVRP and the other for the MDVRP. The same two algorithms were used as partial solvers within each PDG, the hybrid genetic algorithm HGSADC [131] and GUTS, a generalized version of the Unified Tabu Search [26].

Integrator procedures build complete solutions by mixing partial solutions with promising features extracted from the central memories of the PSGs. The resulting complete solutions, together with critical features extracted from the partial solutions, are transmitted to the CSG. Integrators aim for solution quality and diversity, as well as computational efficiency. Because partial solutions are defined by variable fixing rather than deletion, the simplest Integrator consists of selecting high-quality partial solutions (with respect to solution value or the inclusion of particular decision combinations) and passing them directly to the Complete Solver Group. Population-based meta-heuristics make natural integrators, as well as solvers of optimization formulations combining solutions or

solution elements (e.g., set covering for VRP) to yield complete solutions to the problem at hand. The work of [55] belongs to the latter category, proposing particular optimization models for rich VRP settings, which preserve desired critical variables and attributes, present in partial solutions, when selecting and combining routes. Several Integrators can be involved in an ICS meta-heuristic, increasing the diversity of the population of complete solutions. Four Integrators were thus included in the MDPVRP application, the simple one passing good solutions to the CSG, the HGSADC individual enhancement operator (crossover and education), and two of the methods proposed by [55], the first transmitting the attributes for which there was "consensus" in the input solutions, while the second "promoted" them only through penalties added to the objective function. The last three integrators start from pairs of partial solutions randomly selected among the best 25% of the central memory populations of the two PSGs.

The Complete Solver Group includes the ICS central memory together with the SC learning, guidance, and monitoring mechanism. Complete solutions, together with the context information, are received from Integrators. When the CSG includes solvers, these solutions are further enhanced and new ones may be created. Following the concepts described earlier on in this section, knowledge is extracted from the solutions and context data received and created (e.g., the frequency of appearance of each (customer, depot, pattern) triplet in the complete solution set for the MDPVRP, together with the cost of the best solution containing it). This new information is used to build guiding instructions (solutions, patterns of variables to favour or avoid, etc.) sent to the appropriate PSGs according to a monitoring process.

Monitoring is performed by following the evolution of the PSGs (e.g., the number of improving solutions generated during a certain time period) to detect undesired situations, such as loss of diversity in the partial or complete populations, stagnation in improving the quality of the current best solution, awareness that some zones of the search space - defined by particular values for particular decision sets - have been scarcely explored, etc. Whenever one of these situations is detected, the GSC sends guidance instructions to the particular PSG. The particular type of guidance is application specific, but one may inject new solutions or elements, modify the values of the fixed attributes for the PSG to orient its search toward a different area, change the attribute subset under investigation (i.e., change the decomposition of the decision-set attributes), or modify/replace the solution method in a Partial Solver or Integrator. The last two should not occur too frequently. Guidance in [82] takes the form of three solutions, either randomly selected from the complete solution set, or built by the GSC out of promising solution elements with respect to the search history.

Very good results are reported even when compared to the state-of-the-art meta-heuristic for the MDPVRP [82]. The experimental results also indicated that 1) one should use solvers with similar time performances in order to have them contributing reasonably equally to the cooperation; 2) using the PSG central memory as population for

all cooperating GA solvers appears beneficial for short computation runs, while creating solver-specific populations offers better performances for longer runs; 3) embedding good solvers in the CSG enhances slightly the already excellent performance of the ICS parallel meta-heuristic.

10 Conclusions

This chapter presented a synthesis and state-of-the-art survey of the main parallel meta-heuristic ideas, discussing main parallelization concepts and general algorithm-design principles and strategies. The presentation is structured by an enhanced three-dimensional classification of design strategies for parallel meta-heuristics: the number of levels indicating whether decomposition is applied once only or recursively; the decomposition strategy reflecting the sources of parallelism in meta-heuristics, algorithm, search space, or mathematical structure; and the search strategy, given a particular level and decomposition approach, defined by the number of processes controlling the search, the communication and learning mechanism, and the diversity of the individual methods involved and their initial solutions.

Six major classes of parallel meta-heuristics strategies were discussed: low-level decomposition of computing-intensive tasks with no modification to the original algorithm, explicit decomposition of the search space, independent multi-search, as well as synchronous, asynchronous, and knowledge-creating cooperative multi-search. It is noteworthy that this series also reflects the historical sequence of the development of parallel meta-heuristics, which are now acknowledged to make up their own class of meta-heuristics.

The literature survey reveals a rich tapestry of contributions based on one of these strategies in wide array of applications. Two observations strongly indicate, however, that more research is still needed.

First, let us not forget that most optimization problems of interest are complex and computationally difficult. Addressing these problems is increasingly challenging. On the one hand, the dimensions of the instances one faces keep increasing. On the other hand, the problem settings are becoming more complex, illustrated by the network optimization problems involving several layers of interwoven design decisions, the integrated hierarchical hub location and scheduled service network design problems, and the general trend toward explicitly addressing the uncertainty inherently present in most planning and management problems. Harnessing and developing the power of parallel meta-heuristics is crucial to efficiently address realistically dimensioned instances of such problem settings. To achieve these goals requires to study the behaviour of the parallelization strategies within each particular context, and to identify the most appropriate one or combination

thereof.

These developments are linked to the research needs coming from the second observation. Namely, that knowledge and expertise have not progressed equally across parallelization approaches, meta-heuristic designs, and problem settings. It must be emphasized that each of the strategy classes identified in the chapter fulfills a particular type of task and all are needed in some circumstances. Thus, the idea that everything seems to be known regarding low-level parallelization strategies is not true. Most studies on accelerating computing-intensive tasks targeted the evaluation of a population or neighbourhood in classic meta-heuristic frameworks but, as a number of recent studies show, the best strategy to accelerate a local-search procedure may prove less effective when the local search is embedded into a full meta-heuristic or hierarchical solution method. On the other hand, the evolution of computing infrastructure, in particular, the integration of graphical processing units within computing platforms, opens up interesting but challenging perspectives. In both cases, more research is needed to understand their behaviour and identify the most appropriate combination of strategies, particularly low-level and cooperative search, for various meta-heuristics, problem settings, and computing platforms.

Search-space decomposition also seems to have been thoroughly studied, and has been overlooked in the last years, maybe due to the rapid and phenomenal increase in the memory available and the speed of access. Yet, the increased challenges of the evolving combinatorial optimization problem settings, alluded to above, require more research related, in particular, to dynamic search-space decomposition and the combination of cooperative search and search-space decomposition (the Integrative Cooperative Search appears as a first step in this direction).

Asynchronous cooperation, particularly when relaying on memories as communication mechanisms, provides a powerful, flexible, and adaptable framework for parallel meta-heuristics that consistently achieved good results in terms of computing efficiency and solution quality for many meta-heuristic and problem classes. A number of challenging research issues are worth investigating.

A first issue concerns the exchange and utilization of context data gathered locally by cooperating solvers, to infer the status of the global search and generate appropriate guiding instructions. Thus, contrasting and aggregating the information reflecting the various search trajectories of the cooperating solvers may be used to identify regions of the search space that were neglected or over explored. The information could also be used to evaluate the relative performance of the solvers conducting, eventually, to adjust the search parameters of particular solvers or even change the search strategy. So-called "strategic" decision variables or parameters could thus be more easily identified, which could prove very profitable in terms of search guidance.

A related issue concerns the learning processes and the creation of new information out of the shared data. Important questions concern the identification of information that may be derived from the exchanged solutions and its usefulness in inferring the status of the global search, and determining the appropriate guiding information to be sent to solvers. Research in this direction is still at the very beginning but has already proved its worth, in particular in the context of the integrative cooperative methods.

A third broad issue concerns the cooperation of different types of meta-heuristics, as well as the cooperation of meta-heuristics with exact solution methods. The so-called hybrid, hyper, and matheuristic methods represent significant types of method combinations in the sequential optimization field, but very few studies explicitly have targeted associated parallelization strategies. How different methods behave when involved in cooperative search and how the latter behaves given various combinations of methods is an important issue that should yield valuable insights into the design of parallel meta-heuristic algorithms, cooperative ones in particular. A particularly challenging but fascinating direction for cooperative search and ICS is represented by the multi-scenario representation of stochastic optimization formulations, for which almost nothing beyond low-level scenario-decomposition has been proposed yet. The work of [106] on parallel Benders decomposition illustrates the interest of asynchronous cooperation and the challenges of applying them for stochastic network design. Transversal studies comparing the behaviour and performance of particular parallel meta-heuristic strategies over different problem classes, and of different parallel strategies and implementations for the same problem class, would be very valuable in this context, as in the broader field of parallel meta-heuristics.

Acknowledgments

While working on the project, the author was Adjunct Professor, Department of Computer Science and Operations Research, Université de Montréal. The author gratefully acknowledges the financial support provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery grant program, and by the Fonds de recherche du Québec through their infrastructure grants.

References

[1] Aarts EHL, Korst JHM (1989) Simulated Annealing and Boltzmann Machines. John Wiley & Sons, New York, NY

[2] Alba E (ed) (2005) Parallel Metaheuristics: A New Class of Algorithms. John Wiley & Sons, Hoboken, NJ

[3] Alba E, Dorronsoro B (2008) Cellular Genetic Algorithms. Springer, New York

[4] Almeida ALB, Lima J de C, Carvalho MAM (2022) Systematic Literature Review on Parallel Trajectory-based Metaheuristics. ACM Computing Surveys 55(8):1–34

[5] Ammi M, Chikhi S (2015) A Generalized Island Model Based on Parallel and Cooperating Metaheuristics for Effective Large Capacitated Vehicle Routing Problem Solving. Journal of Computing and Information Technology 23(2):141–155.

[6] Arkhipov DI, Wu D, Vu T, Regan AR (2020) A Parallel Genetic Algorithm Framework for Transportation Planning and Logistics Management. IEEE Accesss 8(106506)

[7] Azencott R (1992) Simulated Annealing Parallelization Techniques. John Wiley & Sons, New York, NY

[8] Badeau P, Gendreau M, Guertin F, Potvin JY, Taillard E (1997) A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Transportation Research Part C: Emerging Technologies 5(2):109–122

[9] Banos R, Ortega J, Gil C, Fernandez A, de Toro F (2013) A Simulated Annealing-Based Parallel Multi-Objective Approach to Vehicle Routing Problems with Time Windows. Expert Systems with Applications 40(5):1696–1707

[10] Barr RS, Hickman BL (1993) Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts Opinions. ORSA Journal on Computing 5(1):2–18

[11] Bastos MP, Ribeiro CC (1999) Reactive Tabu Search with Path-Relinking for the Steiner Problem in Graphs. In: Voß S, Martello S, Roucairol C, Osman, IH (eds) Meta-Heuristics 98: Theory & Applications, Kluwer Academic Publishers, Norwell, MA, pp 31–36

[12] Blazewicz J, Moret-Salvador A, Walkowiak R (2004) Parallel Tabu Search Approaches for Two-Dimentional Cutting. Parallel Processing Letters 14(1):23–32

[13] Bortfeldt A, Gehring H, Mack D (2003) A Parallel Tabu Search Algorithm for Solving the Container Loading Problem. Parallel Computing 29(5):641–662

[14] Brodtkorb AR, Hagen TR, Schulz C, Hasle G (2013) GPU Computing in Discrete Optimization. Part I: Introduction to the GPU. EURO Journal on Transportation and Logistics 2(1-2):129–157

[15] Brodtkorb AR, Hagen TR, Schulz C, Hasle G (2013) GPU Computing in Discrete Optimization. Part II: Survey Focussed on Routing Problems. EURO Journal on Transportation and Logistics 2(1-2):159–186

[16] Burke E, Kendall G (eds) (2005) Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques. Springer, New York

[17] Burke E, Kendall G (eds) (2014) Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques, Second Edition. Springer, New York

[18] Burke E, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics, Kluwer Academic Publishers, Norwell, MA, pp 457–474

[19] Cadenas JM, Garido MC, Muñoz E (2009) Using Machine Learning in a Cooperative Hybrid Parallel Strategy of Metaheuristics. Information Sciences 179(19):3255–3267

[20] Cantú-Paz E (2005) Theory of Parallel Genetic Algorithms. In: Alba, E (ed) Parralel Metaheuristics: A New Class of Algorithms, John Wiley & Sons, Hoboken, pp 425–445

[21] Carrabs F, Cerulli R, Mansini R, Moreschini L, Serra D (2024) Solving the Set Covering Problem with Conflicts on Sets: A New Parallel GRASP. Computers & Operations Research 166(106620)

[22] Cavalcante CBC, Cavalcante VF, Ribeiro CC, Souza MC (2002) Parallel Cooperative Approaches for the Labor Constrained Scheduling Problem. In: Ribeiro C, Hansen P (eds) Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Norwell, MA, pp 201–225

[23] Cecilia JM, Garciá JM, Nisbet A, Amos M, Ujaldón M (2013) Enhancing Data Parallelism for Ant Colony Optimization on GPUs. Journal of Parallel and Distributed Computing 73(1):42–51

[24] Chagas GO, Lorena LAN, dos Santos RDC, Renaud J, Coelho LC (2024) A Parallel Variable Neighborhood Search for α-neighbor Facility Location Problem. Computers & Operations Research 165(1406589)

[25] Cordeau JF, Maischberger M (2012) A Parallel Iterated Tabu Search Heuristic for Vehicle Routing Problems. Computers & Operations Research 39(9):2033–2050

[26] Cordeau JF, Laporte G, Mercier A (2001) A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. Journal of the Operational Research Society 52(8):928–936

[27] Crainic TG (2005) Parallel Computation, Co-operation, Tabu Search. In: Rego C, Alidaee B (eds) Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search, Kluwer Academic Publishers, Norwell, MA, pp 283–302

[28] Crainic TG (2008) Parallel Solution Methods for Vehicle Routing Problems. In: Golden, BL, Raghavan, S, Wasil, EA (eds) The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, New York, pp 171–198

[29] Crainic TG (2018) Parallel Meta-Heuristic Search. In: Martí, R, Pardalos, PM, Resende, MGC (eds) Handbook of Heuristics, Springer, New York, pp 809–847

[30] Crainic TG (2019) Parallel Metaheuristics and Cooperative Search. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, Third Edition, Springer, pp 419–451

[31] Crainic TG, Gendreau M (1999) Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. In: Voß S, Martello S, Roucairol C, Osman, IH (eds) Meta-Heuristics 98: Theory & Applications, Kluwer Academic Publishers, Norwell, MA, pp 331–344

[32] Crainic TG, Gendreau M (2002) Cooperative Parallel Tabu Search for Capacitated Network Design. Journal of Heuristics 8(6):601–627

[33] Crainic TG, Gendreau M (2021) Heuristics and Metaheuristics for Fixed-Charge Network Design. In: Crainic TG, Gendreau M, Gendron B (eds) Network Design with Applications in Transportation and Logistics, Springer, Boston, chap 4, pp 91–138

[34] Crainic TG, Hail N (2005) Parallel Meta-Heuristics Applications. In: Alba, E (ed) Parallel Metaheuristics: A New Class of Algorithms, John Wiley & Sons, Hoboken, NJ, pp 447–494

[35] Crainic TG, Toulouse M (1998) Parallel Metaheuristics. In: TG Crainic, G Laporte (eds) Fleet Management and Logistics, Kluwer Academic Publishers, Norwell, MA, pp 205–251

[36] Crainic TG, Toulouse M (2003) Parallel Strategies for Meta-heuristics. In: F Glover F, Kochenberger GA (eds) Handbook in Metaheuristics, Kluwer Academic Publishers, Norwell, MA, pp 475–513

[37] Crainic TG, Toulouse M (2008) Explicit and Emergent Cooperation Schemes for Search Algorithms. In: Maniezzo, V, Battiti, R, Watson, J-P (eds) Learning and Intelligent Optimization, Springer-Verlag, Berlin, Lecture Notes in Computer Science, vol 5315, pp 95–109

[38] Crainic TG, Toulouse M (2010) Parallel Meta-Heuristics. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, Second Edition, Springer, pp 497–541

[39] Crainic TG, Toulouse M, Gendreau M (1996) Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. Annals of Operations Research 63:277–299

[40] Crainic TG, Toulouse M, Gendreau M (1997) Towards a Taxonomy of Parallel Tabu Search Algorithms. INFORMS Journal on Computing 9(1):61–72

[41] Crainic TG, Gendreau M, Hansen P, Mladenović N (2004) Cooperative Parallel Variable Neighborhood Search for the p-Median. Journal of Heuristics 10(3):293–314

[42] Crainic TG, Di Chiara B, Nonato M, Tarricone L (2006) Tackling Electrosmog in Completely Configured 3G Networks by Parallel Cooperative Meta-Heuristics. IEEE Wireless Communications 13(6):34–41

[43] Crainic TG, Li Y, Toulouse M (2006) A First Multilevel Cooperative Algorithm for the Capacitated Multicommodity Network Design. Computers & Operations Research 33(9):2602–2622

[44] Crainic TG, Crisan GC, Gendreau M, Lahrichi N, Rei W (2009) A Concurent Evolutionary Approach for Cooperative Rich Combinatorial Optimization. In: Genetic and Evolutionary Computation Conference - GECCO 2009, July 8-12, Montréal, Canada, ACM, CD-ROM

[45] Crainic TG, Crisan GC, Gendreau M, Lahrichi N, Rei W (2009) Multi-thread Integrative Cooperative Optimization for Rich Combinatorial Problems. In: The 12th International Workshop on Nature Inspired Distributed Computing - NIDISC'09, 25-29 May, Rome, CD-ROM

[46] Crainic TG, Davidović T, Ramljak D (2014) Designing Parallel Meta-Heuristic Methods. In: Despotovic-Zrakic, M, Milutinovic, V, Belic, A (eds) High Performance and Cloud Computing in Scientific Research and Education, IGI Global, Hershey, PA, U.S.A., pp 260–280

[47] Cung VD, Martins SL, Ribeiro CC, Roucairol C (2002) Strategies for the Parallel Implementations of Metaheuristics. In: Ribeiro C, Hansen P (eds) Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Norwell, MA, pp 263–308

[48] Dai C, Li B, Toulouse M (2009) A Multilevel Cooperative Tabu Search Algorithm for the Covering Design Problem. Journal of Combinatorial Mathematics and Combinatorial Computing 68:35–65

[49] Davidović T, Crainic TG (2015) Parallel Local Search to Schedule Communicating Tasks on Identical Processors. Parallel Computing 48:1–14

[50] Delévacq A, Delisle P, Gravel M, Krajecki M (2013) Parallel Ant Colony Optimization on Graphics Processing Units. Journal of Parallel and Distributed Computing 73(1):52–61

[51] Di Chiara B (2006) Optimum Planning of 3G Cellular Systems: Radio Propagation Models and Cooperative Parallel Meta-heuristics. PhD thesis, Dipartimento di ingegneria dell'innovatione, Universitá degli Studi di Lecce, Lecce, Italy

[52] Doerner KF, Hartl RF, Benkner S, Lucka M (2006) Cooperative Savings Based Ant Colony Optimization - Multiple Search and Decomposition Approaches. Parallel Processing Letters 16(3):351–369

[53] Dokeroglu T, E S, Kucukyilmaz T, Cosar A (2019) A Survey on New Generation Metaheuristic Algorithms. Computers & Industrial Engineering 137(106040)

[54] Drias H, Ibri A (2003) Parallel ACS for Weighted MAX-SAT. In: Mira, J, Álvarez, J (eds) Artificial Neural Nets Problem Solving Methods - Proceedings of the 7th International Work-Conference on Artificial and Natural Neural Networks, Lecture Notes in Computer Science, vol 2686, Springer-Verlag, Heidelberg, pp 414–421

[55] El Hachemi N, Crainic TG, Lahrichi N, Rei W, Vidal T (2015) Solution Integration in Combinatorial Optimization with Applications to Cooperative Search and Rich Vehicle Routing. Journal of Heuristics 21(5):663–685

[56] Essaid M, Idoumghar L, Lepagnot J, Brévilliers M (2018) GPU Parallelization Strategies for Metaheuristics: A Survey. International Journal of Parallel, Emergent and Distributed Systems 34(5):497–522.

[57] García-López F, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2003) Parallelization of the Scatter Search for the p-Median Problem. Parallel Computing 29:575–589

[58] García-López F, García Torres M, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2005) Parallel Scatter Search. In: Alba, E (ed) Parallel Metaheuristics: A New Class of Metaheuristics, John Wiley & Sons, Hoboken, NJ, pp 223–246

[59] García-López F, García Torres M, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2006) Solving Feature Subset Selection Problem by a Parallel Scatter Search. European Journal of Operational Research 169(2):477–489

[60] Gehring H, Homberger J (2002) Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows. Journal of Heuristics 8(3):251–276

[61] Gendreau M, Toulouse JY Potvin (2010) Handbook of Metaheuristics, Second Edition. Springer

[62] Gendreau M, Toulouse JY Potvin (2019) Handbook of Metaheuristics, Third Edition. Springer

[63] Gendreau M, Hertz A, Laporte G (1994) A Tabu Search Heuristic for the Vehicle Routing Problem. Management Science 40(10):1276–1290

[64] Gendreau M, Guertin F, Potvin JY, Taillard ÉD (1999) Tabu Search for Real-Time Vehicle Routing and Dispatching. Transportation Science 33(4):381–390

[65] Gendreau M, Laporte G, Semet F (2001) A Dynamic Model and Parallel Tabu Search Heuristic for Real-time Ambulance Relocation. Parallel Computing 27(12):1641–1653

[66] Glover F (1989) Tabu Search – Part I. ORSA Journal on Computing 1(3):190–206

[67] Glover F (1990) Tabu Search – Part II. ORSA Journal on Computing 2(1):4–32

[68] Glover F (1996) Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges. In: Barr R, Helgason R, Kennington J (eds) Interfaces in Computer Science and Operations Research, Kluwer Academic Publishers, Norwell, MA, pp 1–75

[69] Glover F, Kochenberger GA (eds) (2003) Handbook of Metaheuristics. Kluwer Academic Publishers, N.Y.

[70] Glover F, Laguna M (1997) Tabu Search. Kluwer Academic Publishers, Norwell, MA

[71] Goldberg, DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA

[72] Groër C, Golden B, Wasil E (2011) A Parallel Algorithm for the Vehicle Routing Problem. INFORMS Journal on Computing 23(2):315–330

[73] Hidalgo JI, Prieto M, Lanchares J, Baraglia R, Tirado F, Garnica O (2003) Hybrid Parallelization of a Compact Genetic Algorithm. In: Proceedings of the 11th uromicro Conference on Parallel, Distributed and Network-Based Processing, pp 449–455

[74] Hijazi NM, Faris H, Aljarah I (2021) A Parallel Metaheuristic Approach for Ensemble Feature Selection Based on Multi-core Architectures. Expert Systems with Applications 182(115290)

[75] Holland JH (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI

[76] Izzo D, Rucinski M, Ampatzis C (2009) Parallel Global Optimisation Metaheuristics Using an Asynchronous Island-model. In: CEC'09 - IEEE Congress on Evolutionary Computation, IEEE, pp 2301–2308

[77] James T, Rego C, Glover F (2009) A Cooperative Parallel Tabu Search Algorithm for the Quadratic Assignment Problem. European Journal of Operational Research 195(3):810–826

[78] Jin J, Crainic TG, Løkketangen A (2012) A Parallel Multi-Neighborhood Cooperative Tabu Search for Capacitated Vehicle Routing Problems. European Journal of Operational Research 222(3):441–451

[79] Jin J, Crainic TG, Løkketangen A (2014) A Cooperative Parallel Metaheuristic for the Capacitated Vehicle Routing Problem. Computers & Operations Research 44:33–41

[80] Laguna M, Martí R (2003) Scatter Search: Methodology and Implementations in C. Kluwer Academic Publishers, Norwell, MA

[81] Laguna M, Martinez-Gavara A, Perez-Peló S, Resende MGC (2023) 20 Years of Greedy Randomized Adaptive Search Procedures with Path Relinking. DOI: 10.48550/arXiv.2312.12663

[82] Lahrichi N, Crainic TG, Gendreau M, Rei W, Crisan CC, Vidal T (2015) An Integrative Cooperative Search Framework for Multi-Decision-Attribute Combinatorial Optimization. European Journal of Operational Research 246(2):400–412

[83] Le Bouthillier A (2007) Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire. PhD thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, QC, Canada

[84] Le Bouthillier A, Crainic TG (2005) A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. Computers & Operations Research 32(7):1685–1708

[85] Le Bouthillier A, Crainic TG, Kropf P (2005) A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows. IEEE Intelligent Systems 20(4):36–42

[86] Lee SY, Lee KG (1996) Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains. IEEE Transactions on Parallel and Distributed Systems 7(10):993–1007

[87] Ling C, Hai-Ying S, Shu W (2012) A Parallel Ant Colony Algorithm on Massively Parallel Processors and its Convergence Analysis for the Travelling Salesman Problem. Information Sciences 199:31–42

[88] Lopes Silva MA, de Souza SR, Souza MJF, de França Filho MF (2018) Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. Applied Soft Computing 71:433–459

[89] Luque G, Alba E (eds) (2011) Parallel Genetic Algorithyms: Theory and Real World Applications. Springer

[90] Luque G, Luna F, Alba E, Nesmachnow S (2011) Exploring the Accuracy of a Parallel Cooperative Model for Trajectory-Based Metaheuristics. In: Moreno-Díaz R, Pichler F, Arencibia AQ (eds) Computer Aided Systems Theory – EUROCAST 2011 Part I, Springer-Verlag, Lecture Notes in Computer Science, vol 6927, pp 319–326

[91] Maniezzo V, Boschetti MA, Stützle T (2021) Matheuristics - Algorithms and Implementations. Springer

[92] Martí R, Pardalos PM, Resende MGC (eds) (2018) Handbook of Heuristics. Springer, New York

[93] Mehdi M, Loukil L, Bendjoudi A, Melab N (2013) Parallel GPU-accelerated Metaheuristics. In: Couturier R (ed) Designing Scientific Applications on GPUs, Chapman and Hall / CRC, pp 202–216

[94] Melab N, Talbi EG, Cahon S, Alba E, Luque G (2006) Parallel Metaheuristics: Models and Frameworks. In: EL-Ghazali Talbi (ed) Parallel Combinatorial Optimization, John Wiley & Sons, New York, pp 149–162

[95] Melab N, Luong TV, Boufaras K, Talbi EG (2011) Towards ParadisEO-MO-GPU: A Framework for GPU-based Local Search Metaheuristics. In: Cabestany J, Rojas I, Joya G (eds) Advances in Computtational Intelligence - Lecture Notes in Computer Science 6691, Springer, pp 401–408

[96] Michalewicz, Z (1992) Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin

[97] Middendorf M, Reischle F, Schmeck H (2002) Multi Colony Ant Algorithms. Journal of Heuristics 8(3):305–320

[98] Niar S, Fréville A (1997) A Parallel Tabu Search Algorithm For The 0-1 Multidimensional Knapsack Problem. In: 11th International Parallel Processing Symposium (IPPS '97), Geneva, Switzerland, IEEE, pp 512–516

[99] Oduntan IO, Toulouse M, Baumgartner R, Bowman C, Somorjai R, Crainic TG (2008) A Multilevel Tabu Search Algorithm for the Feature Selection Problem in Biomedical Data Sets. Computers & Mathematics with Applications 55(5):1019–1033

[100] Oteiza PP, Rodríguez A, Brignole NB (2018) Parallel Cooperative Optmization Through Hyperheuristics. In: Eden MR, Ierapetritou M, Towler GP (eds) Proceedings of the 13thInternational Symposium on Process Systems Engineering – PSE 2018, Elsevier, pp 201–225

[101] Ouyang M, Toulouse M, Thulasiraman K, Glover F, Deogun JS (2000) Multi-Level Cooperative Search: Application to the Netlist/Hypergraph Partitioning Problem. In: Proceedings of International Symposium on Physical Design, ACM Press, pp 192–198

[102] Ouyang M, Toulouse M, Thulasiraman K, Glover F, Deogun JS (2002) Multilevel Cooperative Search for the Circuit/Hypergraph Partitioning Problem. IEEE Transactions on Computer-Aided Design 21(6):685–693

[103] Pedemonte M, Nesmachnow S, Cancela H (2011) A Survey of Parallel Ant Colony Optimization. Applied Soft Computing 11(8):5181–5197

[104] Penas DR, Henriques D, González P, Doallo R, Saez-Rodriguez J, Banga JR (2017) A Parallel Metaheuristic for Large Mixed-integer Dynamic Optimization Problems, with Applications in Computational Biology. PLoS ONE 12(8):207–218

[105] Polacek M, Benkner S, Doerner KF, Hartl RF (2008) A Cooperative and Adaptive Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. Business Research 1(2):207–218

[106] Rahmaniani R, Crainic TG, Gendreau M, Rei W (2024) An Asynchronous Parallel Benders Decomposition Method for Stochastic Network Design Problems. Computers & Operations Research 162(106459)

[107] Ribeiro CC, Rosseti I (2007) Efficient Parallel Cooperative Implementations of GRASP Heuristics. Parallel Computing 33(1):21–35

[108] Rico-Garcia H, Sanchez-Romero JL, Gomis HM, Rao RzV (2020) Parallel Implementation of Metaheuristics for Optimizing Tool Path Computation on CNC Machining. Computers in Industry 123(103322)

[109] Rios E, Ochi LS, Bœres C, Cœlho VN, Cœlho IM, Faria R (2017) Exploring parallel multi-GPU local search strategies in a metaheuristic framework. Journal of Parallel and Distributed Computing DOI: https://doi.org/10.1016/j.jpdc.2017.06.011

[110] Rochat Y, Taillard ÉD (1995) Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. Journal of Heuristics 1(1):147–167

[111] Sanvicente-Sánchez H, Frausto-Solís J (2002) MPSA: A Methodology to Parallelize Simulated Annealing and Its Application to the Traveling Salesman Problem. In: Coello Coello C, de Albornoz A, Sucar L, Battistutti O (eds) MICAI 2002: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol 2313, Springer-Verlag Heidelberg, pp 89–97

[112] Saviniec L, Santos MO, dos Santos LMR (2004) Pattern-Based Models and a Cooperative Parallel Netaheuristic for High School Timetabling Problems. European Journal of Operational Research 280(3):1064–10,813

[113] Schryen G (2020) Parallel Computational Optimization in Operations Research: A New Integrative Framework, Literature Review and Research Directions. European Journal of Operational Research 287(1):1–18

[114] Schryen G (2024) Speedup and Efficiency of Computational Parallelization: A Unifying Approach and Asymptotic Analysis. Journal of Parallel and Distributed Computing 187(104835)

[115] Schulze J, Fahle T (1999) A Parallel Algorithm for the Vehicle Routing Problem with Time Window Constraints. Annals of Operations Research 86:585–607

[116] Sevkli M, Aydin ME (2007) Parallel Variable Neighbourhood Search Algorithms for Job Shop Scheduling Problems. IMA Journal of Management Mathematics 18(2):117–133

[117] Shami TM, El-Saleh AA, Alswaitti M, Al-Tashi Q, Summakieh MA, Mirjalili S (2022) Particle Swarm Optimization: A Comprehensive Survey. IEEE Access 10:10,031–10,061

[118] Taillard ÉD (1993) Parallel Iterative Search Methods for Vehicle Routing Problems. Networks 23(8):661–673

[119] Taillard ÉD, Gambardella LM, Gendreau M, Potvin JY (1997) Adaptive Memory Programming: A Unified View of Metaheuristics. European Journal of Operational Research 135(1):1–10

[120] Talbi EG (ed) (2009) Metaheuristics: From Design to Implementation. John Wiley & Sons, Hoboken, NJ

[121] Tan Y, Ding K (2016) A Survey on GPU-Based Implementation of Swarm Intelligence Algorithms. IEEE Transactions on Cybernetics 46(9):2168–2267

[122] Tongcheng G, Chundi M (2002) Radio Network Using Coarse-Grained Parallel Genetic Algorithms with Different Neighbor Topology. In: Proceedings of the 4th World Congress on Intelligent Control and Automation, vol 3, pp 1840–1843

[123] Toulouse M, Crainic TG, Gendreau M (1996) Communication Issues in Designing Cooperative Multi Thread Parallel Searches. In: Osman IH, Kelly JP (eds) Meta-Heuristics: Theory & Applications, Kluwer Academic Publishers, Norwell, MA, pp 501–522

[124] Toulouse M, Crainic TG, Sansó B, Thulasiraman K (1998) Self-Organization in Cooperative Search Algorithms. In: Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, Omnipress, Madisson, WI, pp 2379–2385

[125] Toulouse M, Crainic TG, Sansó B (1999) An Experimental Study of Systemic Behavior of Cooperative Search Algorithms. In: Voß S, Martello S, Roucairol C, Osman IH (eds) Meta-Heuristics 98: Theory & Applications, Kluwer Academic Publishers, Norwell, MA, pp 373–392

[126] Toulouse M, Thulasiraman K, Glover F (1999) Multi-Level Cooperative Search: A New Paradigm for Combinatorial Optimization and an Application to Graph Partitioning. In: Amestoy P, Berger P, Daydé M, Duff I, Frayssé V, Giraud L, Ruiz D (eds) 5th International Euro-Par Parallel Processing Conference, Lecture Notes in Computer Science, vol 1685, Springer-Verlag, Heidelberg, pp 533–542

[127] Toulouse M, Crainic TG, Thulasiraman K (2000) Global Optimization Properties of Parallel Cooperative Search Algorithms: A Simulation Study. Parallel Computing 26(1):91–112

[128] Toulouse M, Crainic TG, Sansó B (2004) Systemic Behavior of Cooperative Search Algorithms. Parallel Computing 30(1):57–79

[129] Vallada E, Ruiz R (2009) A Cooperative Metaheuristics for the Permutation Flowshop Scheduling Problem. European Journal of Operational Research 193(2):365–376

[130] Van Luong T, Melab N, Talbi EG (2013) GPU Computing for Parallel Local Search Metaheuristic Algorithms. IEEE Transactions on Computers 62(1):173–185

[131] Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A Hybrid Genetic Algorithm for Multi-Depot and Periodic Vehicle Routing Problems. Operations Research 60(3):611–624