Chapter 13
Parallel Metaheuristics and Cooperative Search

Teodor Gabriel Crainic

Abstract The chapter presents a general view of parallel metaheuristics for optimization. It recalls the main concepts and strategies in designing parallel metaheuristics and identifies trends and promising research directions. The focus is on cooperation-based strategies, which display remarkable performances, in particular strategies based on asynchronous exchanges and the creation of new information out of exchanged data to enhance the global guidance of the search.

13.1 Introduction

The development of metaheuristics that take advantage of parallel computing aims for two major goals. The first is common to all parallel computing development efforts: 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. The second is proper to approximate solution methods, e.g., simple heuristics, metaheuristics, and matheuristics, and it concerns the method's so-called robustness, that is, its capability to offer a consistently high level of performance over a wide variety of problem settings and instance characteristics. In appropriate settings, e.g., the cooperative multi-search strategies (Sect. 13.6), parallel metaheuristics proved to be much more robust than sequential versions. Moreover, they also generally require less extensive, and expensive, parameter-calibration efforts.

T. G. Crainic (✉)
CIRRELT - Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Montréal, QC, Canada

School of Management, Université du Québec à Montréal, Montréal, QC, Canada
e-mail: TeodorGabriel.Crainic@cirrelt.net

M. Gendreau, J.-Y. Potvin (eds.), *Handbook of Metaheuristics*,
International Series in Operations Research & Management Science 272,
https://doi.org/10.1007/978-3-319-91086-4_13

The objective of this chapter is to present an overview of the parallel metaheuristics field in a unified manner. It thus recalls the main concepts and general strategies for the design of parallel metaheuristics, including the main approaches to instantiate them for neighborhood- and population-based metaheuristics. Note that the chapter focuses on the *design* of the new class of algorithms parallel metaheuristics create, and, thus, not on their implementation on particular computing architectures. We do, however, identify new trends, challenges, and opportunities that some of the new computing-platform developments bring to the field. We complete the chapter with a number of major open questions and research challenges.

As the chapter follows and updates a previous publication [27], it focuses on more recent developments (typically, from 2005 to 2017) and, in particular, on cooperation-based strategies, which display remarkable performances for a broad range of optimization problems. In addition to the references provided in the following sections, the reader may consult a number of surveys, taxonomies, and syntheses, e.g., [1, 2, 19, 20, 23, 27, 35, 71, 80, 95].

The chapter is organized as follows. Section 13.2 is dedicated to a general discussion of the potential for parallel computing in metaheuristics, a brief description of performance indicators for parallel metaheuristics, and the taxonomy used to structure the presentation. Section 13.3 addresses strategies focusing on accelerating computing-intensive tasks without modifying the basic algorithmic design. Methods based on the explicit separation of the search space are treated in Sect. 13.4, while strategies based on the simultaneous exploration of the search space by several independent metaheuristics constitutes the topic of Sect. 13.5. Cooperation principles and strategies are discussed in Sect. 13.6 and are detailed in Sects. 13.6.1, 13.6.2, and 13.7. We conclude in Sect. 13.8.

13.2 Metaheuristics and Parallelism

This section is dedicated to a brief overview of the main potential sources for parallel computing in metaheuristics, followed by a discussion of performance indicators for parallel metaheuristics. The section concludes with the criteria used in this paper to describe and classify parallelization strategies for metaheuristics.

13.2.1 Sources of Parallelism

Parallel/distributed/concurrent computing means that several processes work simultaneously on several processors addressing a given problem instance and aiming to identify the best (or a) 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. According to how "small" or "large" the tasks are in terms of algorithm 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 structure. *Functional parallelism* (Sect. 13.3) corresponds to the first case, according to which computing-intensive parts of the algorithm are decomposed into a number of tasks (processes), working on the same data or on dedicated parts of the data, are allocated to different processors and run in parallel, possibly exchanging information. The concurrent execution of the innermost loop iterations, e.g., evaluating neighbors, computing the fitness of individuals, or having ants forage concurrently, provides the main source of functional parallelism for metaheuristics This is often also the only source of readily available parallelism in metaheuristics, the execution of most other steps in the algorithm depending on the status of the search, e.g., what has been performed so far and the values of the decision variables, which requires either the computation of the previous steps to be completed, or the synchronization of computations; and synchronization generally yields significant delays, which may make such parallel computation non relevant. Traditionally, functional parallelism was therefore interesting as a low-level component 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. The rapid development in the utilization of the *graphical processing units* (*GPU*), ubiquitous within most computers, is changing this statement as very impressive reductions in computing times may be obtained (Sect. 13.3).

Search *space separation*, constitutes a second major class of parallel strategies. We find under this umbrella the two other cases mentioned above, i.e., the search space and the problem structure. The general idea is to decompose the problem domain, or the associated search space (for brevity reasons and without loss of generality, the latter term is used in this chapter), and to address the problem on each of the resulting components using a particular solution methodology. Indeed, there are no data dependencies between the evaluation functions of different solutions and, thus, these may be computed in parallel. Moreover, theoretically, the parallelism in the solution or search space is as large as the space itself when a processor is assigned to each solution. Obviously, the latter strategy is not practical and the search space is separated into subspaces assigned to different processors. Such a separation still leaves a search space for each processor too large for explicit enumeration, however, and, thus, an exact or heuristic search method is required to implicitly explore it.

Space separation is exploited in many of the strategies described in this chapter, but raises a number of issues with respect to an overall metaheuristic search strategy, e.g., how to separate; how to control an overall search conducted separately on several components of the original space; how to create a complete solution out of the ones obtained on each component; how to allocate resources for an efficient exploration avoiding, for example, regions with poor-quality solutions. The answers to these questions yield several classes of algorithms described in the following sections. These may be grouped, however, into two main approaches: *domain decomposition* and *multi search*. The former explicitly separates the space yielding a number of subproblems to be addressed simultaneously, their solutions being then combined into solutions to the original problem, while the latter performs the separation implicitly, through concurrent explorations by several methods, named *solvers* in the following, which may exchange information or not.

The metaheuristic or exact solvers involved in a multi-search metaheuristic may address either the complete problem at hand, or explore partial problems defined by decomposing the initial problem through mathematical programming or attribute-based heuristic approaches. In the former case, the decomposition method implicitly defines how a complete solution is built out of partial ones. In the latter case, some processors 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. Multi-search strategies, particularly those based on cooperation principles, are at the core of the most successful developments in parallel metaheuristics and are the object of the later sections of this chapter.

13.2.2 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 that run 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 [5] for a detailed discussion of this issue, including additional performance measures.

Speedup measures are more difficult to define when the optimal solution is not guaranteed or the exact method is stopped before optimality is reached, which is obviously also the case for metaheuristics. Moreover, most strategies to build parallel metaheuristics yield solutions that are different in value, composition, or both from those of the sequential versions (when they exist). Hence, an equally important objective for parallel metaheuristics is to what extend they outperform their sequential counterparts in terms of solution quality and, ideally, computational efficiency. 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. Robustness with respect to a problem setting is meant in the sense of providing "equally" good solutions to a large and varied set of problem instances, without excessive calibration, neither during initial development, nor when addressing new instances. Multi-search methods, particularly those based on cooperation, generally display a behavior different from those of the sequential methods involved,

offering enhanced performances compared to sequential methods and other parallelization strategies in terms of solution quality and method robustness (see [24, 25] for a discussion of these issues). They are thus generally acknowledged as proper metaheuristics [1].

13.2.3 Parallel Metaheuristics Strategies

We adopt the classification of [23], generalizing that of [29], to describe the different parallel strategies for metaheuristics. This classification is sufficiently general to encompass the principal parallel metaheuristic classes, while avoiding a level of detail incompatible with the scope and dimension limits of the chapter.

The three dimensions of the classification define how the global problem-solving process is controlled, how information is exchanged among processes and how, eventually, new information is created, and the diversity of searches involved, respectively. Table 13.1 synthesizes the dimensions and categories of the classification, which are now detailed.

The first dimension, *Search Control Cardinality*, specifies whether the global search is controlled by a single process or by several processes that may collaborate or not. The two categories are identified as *1-control (*1C*)* and *p-control (*pC*)*, respectively.

The second dimension, relative to the type of *Search Control and Communications*, addresses the issue of information exchanges and the utilization of the exchanged information to control or guide the search. In parallel computing, one generally refers to *synchronous* and *asynchronous* communications. In the former case, all concerned processes stop and engage in some form of communication and information exchange at moments (number of iterations, time intervals, specified algorithmic stages, etc.) exogenously determined, either hard-coded or imposed by a control (master) process. In the latter case, each process is in charge of its own search, as well as of establishing communications with other processes, and the global search terminates once all individual searches stop. Four categories are defined to reflect the quantity and quality of the information exchanged and shared, as well as the additional knowledge derived from these exchanges (if any); two for synchronous settings, *Rigid (RS)* and *Knowledge Synchronization (KS)*, and, symmetrically, two for asynchronous strategies, *Collegial (C)* and *Knowledge Collegial (KC)*.

Table 13.1 The parallel metaheuristics taxonomy

Dimension	Categories			
Control Cardinality	*1C*		*pC*	
Control and Communications	*RS*	*KS*	*C*	*KC*
Differentiation	*SPSS*	*SPDS*	*MPSS*	*MPDS*

More than one solution method or variant (e.g., with different parameter settings) may be involved in a parallel metaheuristic and such solvers may be (meta-)heuristics or exact solution methods. The third dimension thus indicates the *Search Differentiation* or diversity: do solvers start from the same or different solutions, and are they the same or not? Note that one characterizes two solvers as "different" even when based on the same methodology (e.g., two tabu searches or genetic algorithms) if they use different search strategies in terms of components (e.g., neighborhoods 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 neighborhood-based, single-solution methods, while "population" is used for population-based ones.

13.3 Low-Level Parallelization Strategies

Functional-parallelism-based strategies, exploiting the potential for task decomposition within the inner-loop computations of metaheuristics, aim to accelerate the search, without modifying the algorithmic logic, the search space and behavior of the sequential metaheuristic. Hence the label "low level" often associated with such strategies. Typically, the exploration is initialized from a single solution or population, and proceeds according to the sequential metaheuristic logic, while a number of intensive-computation steps are decomposed and simultaneously performed by several processors.

Most low-level parallel strategies belong to the 1C/RS/SPSS class and are usually implemented according to the classical *master-slave* parallel programming model. A "master" program executes the (1-control) sequential metaheuristic, separating and dispatching computation-intensive tasks to be executed in parallel by "slave" programs. Slaves perform the tasks and return the results to the master which, once all the results are in, resumes the normal logic of the sequential metaheuristic. The master thus has complete control on the algorithm execution; it decides the work allocation for all other processors and initiates communications. No communications take place among slave programs.

The neighborhood-evaluation procedure of the local search heuristics, used alone or as component of neighborhood- or population-based metaheuristics (implementing advanced "schooling" for offspring in the latter case) is generally targeted in 1C/RS/SPSS designs. The master groups the neighbors into tasks and sends them to slaves. Each slave then executes the exploration/evaluation procedure on its respective part of the neighborhood, and sends back the best, or first improving, neighbor found. The master waits for all slaves to terminate their computations, selects the best move and proceeds with the search. The appropriate granularity of the decomposition, 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. Thus, for example, [38] discusses several decomposition policies for the permutation-based local search neighborhood applied to the scheduling of dependent tasks on homogeneous processors, and shows that the uniform partition usually called upon in the literature is not appropriate in this context characterized by neighborhoods of different sizes. The authors also show that a fixed coarse-grained non-uniform decomposition, while offering superior results, requires calibration each time the problem size or the number of processors varies. The best performing strategy, called *dynamic fine-grained* by the authors, defines each neighbor evaluation as a single task, the master dynamically dispatching these on a first-available, first-served basis to slave processors as they complete their tasks. The strategy partitions the neighborhood into a number of components equal to the number of available processors, but of unequal size with a content dynamically determined at each iteration. The dynamic fine-grained strategy provides maximum flexibility and good load balancing, particularly when the evaluation of neighbors is of uneven length. The uniform distribution appears more appropriate when the neighbor evaluations are sensibly the same, or when the overhead cost of the dynamic strategy for creating and exchanging tasks appears too high.

Similar observations may be made regarding population-based metaheuristics. In theory, all genetic-algorithm operators may be addressed through a 1C/RS/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 parallelizing, 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 1C/RS/SPSS parallelism for genetic-evolutionary methods, usually implemented using the master-slave model.

The 1C/RS/SPSS parallelism for ant-colony 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/RS/SPSS strategy according to the master-slave model (e.g., [41] and references herein). The master builds tasks consisting of small colonies of one or a few ants, and distributes them to the available processors. Slaves perform their construction heuristic and return their solution(s) to the master, which updates the pheromone matrix, returns it to the slaves, and so on. To further speed up computation, the pheromone update can be partially computed at the slave level, each slave 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 metaheuristics, it is outperformed by more advanced multi-search methods.

Scatter search and path relinking implement different evolution strategies, where a restricted number of elite solutions are combined, the result being enhanced through a local search or a full-fledged metaheuristic, usually neighborhood-based. Consequently, the 1C/RS/SPSS strategies discussed above apply straightforwardly as in [46–48] for the *p*-median and the feature-selection problems. A different 1C/RS/SPSS strategy for scatter search may be obtained by running concurrently the combination and improvement operators on several subsets of the reference set. Here, the master generates tasks by extracting a number of solution subsets, which are sent to slaves. Each slave then combines and improves its solutions, returning its results to the master for the global update of the reference set. Each subset sent to a slave may contain the exact number of solutions required by the combination operator or a higher number. In the former case, the slave performs an "iteration" of the scatter search algorithm [46–48]. 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 slaves.

To conclude, low level, 1-control parallel strategies are particularly attractive when neighborhoods or populations are large, or the neighbor or individual evaluation is costly. Computing time gains may then be obtained, as illustrated by many early contributions discussed in the surveys indicated in the Introduction. Even more impressive gains may be obtained by taking advantage of the current computing platforms integrating multi-core central processing units (CPUs—the "traditional" processor) and graphical processing units (GPUs) enhanced with data streaming, i.e., hardware data parallelism providing the means for each processor to perform the same task on different parts of the distributed data (e.g., [10, 11]). This hardware technology offers the possibility of extensive very low-level parallelization reminiscent of the work performed for the massively parallel computers of the late eighties. The neighborhood evaluation in local search heuristics, the fitness evaluation of evolutionary methods, and the evolution of individuals in swarms may clearly benefit from such a hardware-oriented parallelization, spectacular speedups having been observed (e.g., [10, 11, 14, 39, 72, 97, 107]). 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. Second, there is also the need to examine the sequential and parallel metaheuristic designs to identify and valuate where this technology would bring the most benefits, besides those already identified. The work of [84] 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 1C/RS/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 GPUs.

13.4 Domain Decomposition

We group under this title the strategies that separate the search space explicitly. The basic idea is intuitively simple and appealing: separate the search space into smaller subspaces, address the resulting subproblems by applying the sequential metaheuristic on each subspace, 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 metaheuristic.

Regarding the separation type, the resulting subspaces may constitute a *partition* of the complete space (disjoint subspaces, their union being the full space), or a *cover* allowing a certain amount of overlap among the subspaces. 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., neighborhood moves or individual crossovers.

The separation may be obtained by identifying a subset of variables, and corresponding constraints, eventually, and by discarding or fixing the other variables and constraints, the goal being to obtain smaller, easier to address subproblems. Note that it is not always possible, even desirable, to discard. Thus, if one may easily discard the customers in a Vehicle Routing Problem (VRP) that do not belong to a given subspace (the depot must be included in each subspace) and solve the resulting partial VRPs separately, doing the same is much more difficult to implement when considering the commodities and arcs of a Multicommodity Capacitated Network Design problem (MCND). Separation by variable fixing (and projection of the corresponding constraints) appears more flexible as one still works on smaller subproblems, but considering the complete vector of decision variables, some of which are fixed. It is also a more general approach, as we find it in advanced cooperative search methods, e.g., [64].

Strict partitioning restricts the solvers to their subspaces, resulting in part of the search space being unreachable and the loss of exploration quality for the parallel metaheuristic. 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 in strict partitioning, 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 provides also 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-slave 1C/RS schemes, with MPSS or MPDS search differentiation. The master process determines the separation and sends partial subsets (or information to define them out of the initial space—this reduces the communication overhead) to slaves, synchronizes them and

collects their solutions, reconstructs complete solutions, modifies the separation, and determines when stopping conditions are met. Slaves 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 (e.g., [52] for real-time ambulance fleet management), a full-fledged metaheuristic being generally used on each subspace.

Explicit space separation may also be performed in a pC, collegial decision-making, framework with MPSS or MPDS search-differentiation. The separation in a pC/KS strategy is collegially decided and modified through information-exchange phases (e.g., round-robin or many-to-many exchanges) activated at given synchronization points. The KS label comes from exchanging not only the best solutions in each subspace (e.g., routes in a VRP), but also from the so-called context information (e.g., un-serviced customers and empty vehicles in a VRP [91]) that is used to modify the separation. In the initial proposition by [91] for the VRP (simulated on a sequential machine), the customer set was partitioned, vehicles were allocated to the resulting regions, each subproblem was solved by an independent tabu search, synchronization occurred after a number of iterations that varied according to the total number of iterations already performed, and exchanges took place between adjacent processors (corresponding to neighboring regions). The method allowed at the time to address successfully a number of problem instances, but the synchronization inherent in the design of the strategy hindered its performance. A parallel ant-colony approach for the VRP based on this idea was presented in [41] with good speedup results when the number of customer increased.

Domain decomposition methods induce different search behavior and solution quality compared to those of the sequential metaheuristic. Such methods appear increasingly needed as the dimensions of the contemplated problem instances continue to grow. Given the increased complexity of the problem settings, work is also required on how to best combine search-space separation and the other parallelization strategies, cooperation in particular. The Integrative Cooperative Search [64] is a step in this direction (see Sect. 13.7).

13.5 Independent Multi-Search

We dedicate a section to the *Independent multi-search* as it was among the first parallelization strategies 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 sequential search) by initiating simultaneous solvers from different initial points (with or without different search strategies). It thus parallelizes the classical multi-start strategy by performing several searches

simultaneously on the entire search space, starting from the same or from different initial solutions, and selecting at the end the best among the best solutions obtained by all searches. Independent multi-search methods thus belong to the pC/RS class of the taxonomy. 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 [7, 90, 92, 98]. The surveys identified in the Introduction describe numerous contributions of applying the pC/RS independent multi-search strategy to a variety of combinatorial optimization problems.

Independent multi-search offers an easy access to parallel metaheuristic 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.

13.6 Cooperative Search

Cooperative multi-search has emerged as one of the most successful metaheuristic methodologies to address hard optimization problems (e.g., [1, 18, 19, 23, 26, 27, 94, 96]). While independent multi-search strategies seek to accelerate, compared to sequential search, the exploration toward a better solution by initiating simultaneous searches from different initial points, cooperative search strategies 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 sharing and, eventually, creation of new information out of the exchanged data (Sect. 13.7), yields in many cases a collective output with better solutions than a parallel independent search.

Cooperative-search strategies are thus defined by the solver components engaged in cooperation, their interaction mechanism, and the nature of the information shared. The solvers define trajectories in the search space from possibly different initial points or populations, by using possibly different search strategies (including, possibly exact methods). The information-sharing cooperation mechanism specifies how these independent solvers interact, how the exchanged information is used globally (if at all), and how each solver acts on the received information, using it within its own search and, thus, transforming it before passing it to other solvers.

The information-sharing cooperation mechanism specifies how these independent solvers interact, the global search behavior of the cooperative parallel metaheuristic emerging from the local interactions among them, which makes it a "new"

metaheuristic in its own right [26]. The similarity between this behavior and that of systems where decisions emerge from interactions among autonomous and equal "colleagues" has inspired the name *collegial control* associated with cooperative-search strategies in the taxonomy used in this chapter. The various cooperation mechanisms proposed in the literature are described in the next sections.

Exchanged information must be *meaningful* and exchanges must be *timely*. The goals are (1) to improve the performance of the receiving solvers, and (2) to create as much as possible a global, "complete" image of the status of the cooperative search to enable guiding it, through its 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 was proposed in [100]. The list is still relevant when designing cooperative parallel strategies: What information is exchanged? Between what processes is it exchanged? When is information exchanged? How is it exchanged? How is the imported data used? Implicit in their taxonomy and explicitly stated in later papers, the issue of whether the information is modified during exchanges or whether new information is created completes this list.

"Good" solutions are 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. It was observed, however, that sending out all current-best solutions a solver identifies is often counter productive, particularly when the solver performs a series of improving moves or generations, as solutions are generally "similar" (particularly for neighborhood-based procedures), and the receiving solvers 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 sending 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 memories of tabu search), the impact of particular moves on the search trajectory (e.g., the scores of the moves of large adaptive neighborhood search), population diversity measures, individual resilience across generations, etc. A limited number of studies indicate the interest of context-information exchanges (see Sect. 13.7), 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 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 metaheuristic literature. The term *memory* is used in this chapter.

Centralized-memory mechanisms have been used in most parallel metaheuristic contributions. They receive, eventually process, and post information received from all 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 *communication 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. The projection of this graph on the physical interconnection topology of the parallel computer executing the parallel program is generally part of the implementation design.

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 are identified as *synchronous* and *asynchronous* communications, respectively, 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 [101, 102, 104, 105].

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 that exchange current-best solutions only.

It is thus important to equip the cooperation mechanisms with diversification capabilities, e.g., probabilistic or diversity-driven selection of exchanged solutions as proposed in [108], or creation of new solutions and guidance information [64].

Second, the main principles of cooperative parallelization are the same for neighborhood- and population-based metaheuristics, 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 metaheuristic class. The next two subsections discuss the classic synchronous and asynchronous strategies, while the advanced methods based on creation of new information out of the shared one are the topic of Sect. 13.7.

13.6.1 pC/KS Synchronous Cooperative Strategies

Synchronous cooperation follows a p-control (pC), knowledge synchronous (pC/K) strategy, with any of the SPDS, MPSS or MPDS search differentiation approaches. All participating solvers 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 exogeneously to all solvers (e.g., number of iterations from the last synchronization point), or detected by an a priori designated solver.

The goal of synchronous cooperative strategies is 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. This goal is generally only partially attained, however, even though these strategies have generally outperformed the sequential versions as well as simpler parallelization strategies. Moreover, synchronization results in significant time inefficiencies as communications are initiated only when the slowest search thread is ready to start. Asynchronous information sharing thus intuitively appears more promising and, indeed, cooperation based on asynchronous exchanges, described in the following sections, generally outperformed synchronous methods. Consequently, few contributions relying on synchronous cooperation were proposed in recent years.

We therefore restrict this section to recalling the main concepts used in synchronous cooperation, some of which found their way into more advanced strategies, encouraging interested readers to consult the surveys indicated in the Introduction for details and references.

Synchronization may use a complete communication graph or a more restricted, less densely connected communication topology (e.g., ring, torus, and grid graph). *Global* exchanges of information among all solvers take place in the former case, while information follows a *diffusion* process through direct, *local*, exchanges among neighboring processes in the latter.

In a restricted view of the concept, a number of proposed pC/KS cooperative search metaheuristics based on global exchanges use a designated *communication*

master process, which may or not include one of the participating solvers. The communication master manages the synchronization mechanism in a master-slave 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 (e.g., [45, 82]). For coarse-grained island implementations of cooperating genetic methods, synchronization means the communication master initiates the *migration* operator to exchange among the independent populations the best or a small group of some of the best individuals in each [36, 89]. For ant-colony systems, this strategy divides the colony into several sub-colonies individually assigned to solvers, the master updating the pheromone matrix, and starting a new search phase, based on the received solver results [42]. A more sophisticated approach was proposed in [76], where the master dynamically adjusted the search parameters of cooperating tabu searches 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 more sophisticated cooperation mechanisms.

A truer global pC/KS cooperative scheme empowers solvers to initiate synchronization. Once it reaches a pre-determined status, a solver thus 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 genetic-based evolutionary methods fall under this category, where migration operators are applied at regular intervals [44, 59] ([59] implementing a hierarchical method with the fitness computation performed at the second level through a master-slave implementation; the overhead due to the parallelization of the fitness became significant for larger numbers of processors). For ant-colony application, where each colony evolves its own pheromone matrix, global synchronization means that, after a fixed number of iterations, colonies exchange elite solutions that are used to update the pheromone matrix of the receiving colony [73, 74].

Synchronization based on global exchanges of information assumes that making available to all solvers the entire information shared will result in superior performances. Other than the often excessive communication overhead, the main drawback is that solvers relying heavily on the same information end up by exploring the same regions of the search space, resulting in loss of diversity and efficiency. Two approaches have been proposed to overcome this drawback.

First, do not share and use uniquely the local best solutions, as shown in the pC/RS/MPDS iterated tabu search proposed for the VRP in [16]. In this work, solvers synchronized after a number of consecutive iterations without improvement within the individual improvement phases. Synchronization involved the exchange of the good solutions obtained by the solvers and, then, each individual solver built

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. In such strategies, direct communications at synchronization points are possible only with neighboring solvers, i.e., with nodes adjacent in the sparse communication 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 modification by the intervening solvers. This idea was less explored compared to the global-exchange strategy, even though synchronous cooperative mechanisms based on local exchanges and diffusion have a less negative impact on the diversity of the search-space exploration and have yielded good results (e.g., [74, 99]).

13.6.2 pC/C Asynchronous Cooperative Strategies

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

A cooperation strategy is asynchronous when programs initiate cooperation activities according to their own internal logic, without coordination with other solvers or memories. 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 failed to improve the quality of its best solution for a certain number of iterations.

Asynchronous communications provide the means to build cooperation and information sharing among search threads without incurring the overheads associated with synchronization. They also bring adaptability to cooperation strategies, to the extent that the parallel cooperative metaheuristic may more easily react and dynamically adapt to the exploration of the search space than independent or synchronous search. These benefits come with potential issues one must care for. For example, the information related to the global search that is available when a solver must take an action may be less "complete" than in a synchronous environment. On the other hand, too frequent data exchanges, combined with simple acceptance rules for incoming information, may induce an erratic solver behavior, the corresponding search trajectories becoming similar to random walks. Hence the interest for applying information-sharing based on quality, meaningfulness, and parsimony principles [28, 29, 100].

Asynchronous cooperative strategies follow either pC/C or pC/KC collegial principles, the main difference between the two being that in the latter "new" knowledge is inferred on the basis of the information exchanged between solvers; pC/KC strategies are addressed in the next section.

In most pC/C asynchronous strategies in the literature, the shared information corresponds to a locally improving solution or individual(s), the most successful contributions sharing local optima only. The principles mentioned above also resulted in mechanisms to diversify the shared information [28]. Thus, always selecting the best available solution out of an elite set of good solutions, sent by potentially different solvers, proved less efficient in terms of quality of the final solution than a strategy that selected randomly, but biased by quality, among the same elite set.

When to initiate and perform cooperation activities, as well as how to use the incoming information is particular to each type of metaheuristic. Most strategies proposed in the literature follow the same idea, however, to send and request information jointly. There is no need to do this, however, even though it can decrease the amount of communication. It may thus be interesting for neighborhood-based methods to make available right away their newly found local optima or improved overall solutions, and not wait for the algorithmic step where examining external information is appropriate. Similarly, population-based methods could 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 of external information, it may be based on a pre-fixed number of iterations, but this approach should be restricted to metaheuristics 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, e.g., diversification in tabu search, change of neighborhood in variable neighborhood search, and complete or partial re-generation of population in population-based metaheuristics, which involves the choice or modification of the current solution to initiate a new phase. 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; in may be used to initiate a diversification phase, to modify the search trajectory through a combination with a "local" solution, or to modify the solver behavior in the long run through an insertion in an elite set or population. As already mentioned, however, one tries to avoid frequent imports followed by a replacement of the current solution or population, which will result in a random search.

Direct and indirect exchange pC/C strategies may be used with any metaheuristic. Historically, however, most genetic-based evolutionary asynchronous cooperative metaheuristics relied on direct communications over complete communication graphs [12]. These methods generally implement a coarse-grained island model, migration being triggered by conditions within individual populations, selected migrant individuals being directed toward either all other populations or a dynamically-selected subset. The work in [106] illustrates this approach, where migration is initiated by an island that identified a new best solution, which it sends to all other islands. The migrant individual is accepted by the solver of another island only when different from the local population and better than the worst individual in

that population. We also mention the work of [60] 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/C models. The parallelization of ant-colony methods may use the same approach, where partitions of the initial colony play the role of islands. The contribution of [70] is interesting in this context for a 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.

Notice that complete communication graphs are not compulsory. Indeed, one could use particular graphs and information-diffusion processes tailored to the problem at hand. Yet, despite encouraging results, e.g., [88] proposing VNS pC/C strategies over uni and bidirectional ring topologies, too few experiments have been reported yet.

Historically, the sharing of information in most asynchronous cooperative search strategies outside the genetic-evolutionary community is based on some form of indirect communications through a centralized device—data repository/processor -, often called *central memory* [18, 28, 29]. 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 the other cooperating solvers. Classical retrieval mechanisms are based on random selection, which may be uniform or biased to favor 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 metaheuristic classes. To the best of our knowledge, the authors in [28] were 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. Their method, where individual tabu searches sent to the memory their local-best solutions when improved, and imported a solution selected probabilistically biased by rank before engaging in a diversification phase, 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 [22].

pC/C with some form of central memory were proposed for a variety of problems, including cutting [8], container loading [9], labor-constrained scheduling [13], and VRP with time windows (VRPTW) [66]. On the other hand, several studies focused

on pC/C strategies with some form of central memory for particular classes of metaheuristics like simulated annealing (e.g., [4, 68, 86], the latter for multi-objective problem settings), VNS (e.g., [30, 81], 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 [83], and tabu search with memory hosting a reference set and long-term global memories built on short-term local memories sent by solvers [61].

Notice that cooperating solvers need not belong to the same metaheuristic class. The next section will show several examples where different metaheuristics collaborate within pC/KC strategies. We find, in the classical pC/C case, contributions following the same broad strategy described above when calling sequentially on metaheuristics belonging to different types. The two-phase approach of [49] for the 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 tabu search to minimize the total distance traveled. A somewhat different two-phase pC/C parallel strategy was proposed in [6] for the Steiner problem, where each phase, using reactive tabu search and path relinking, respectively, implemented the pC/C asynchronous central memory strategy, all processes switching from the first to the second phase simultaneously.

Multi-level cooperative search proposes a different pC/C asynchronous cooperative strategy based on controlled diffusion of information [103]. 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", the first-level solver working on the complete original problem. It communicates exclusively with the two solvers directly above and below, that is, at higher and lower aggregation levels, respectively. The local memory is used to receive the information coming from the immediate neighbors and to access it at moments dynamically determined according to the internal logic of the solver. 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 have been obtained for various problem settings including graph and hypergraph partitioning [78, 79], network design [32], feature selection in biomedical data [77], and covering design [37]. It is noteworthy that one can implement multi-level cooperative search using a central memory by adequately defining the communication protocols. Although not yet fully defined and tested, this idea is interesting as it opens the possibility of richer exchange mechanisms combining controlled diffusion and general availability of global information.

The central-memory pC/C asynchronous cooperation strategy is generally offering very good results, yielding high-quality solutions. It is 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 exchanged information (kept in the memory) has brought the question of whether one could design more advanced cooperation mechanisms taking advantage of the information exchanged among cooperating solvers. The pC/KC strategies described in the next section are the result of this area of research.

13.7 pC/KC Cooperation Strategies: Creating New Knowledge

Cooperation, particularly in the central-memory asynchronous form, offers many possibilities for algorithm development. Particularly noteworthy are the flexibility in terms of the different metaheuristic 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 neighborhood-based improving metaheuristics, evolve populations of solutions, or perform post-optimization procedures on solutions in memory.

The study reported in [21] illustrates the interest of these ideas. The authors combined a genetic solver and several solvers executing the pC/C tabu search for the multicommodity location-allocation problem with balancing requirements of [28]. The tabu searches were aggressively exploring the search space, building the elite solution set in the central memory, while the genetic method contributed toward increasing the diversity, and hopefully the quality, of the solutions in the central memory, which the cooperating tabu search methods would then import. The genetic method was launched once a certain number of elite solutions identified by the tabu searches were recorded in the central memory, using this memory as initial population. Asynchronous migration subsequently transferred the best solution of the genetic pool to the central memory, as well as solutions of the central memory toward the genetic population. This strategy did perform well, especially on larger instances. It also yielded an interesting observation: while the best overall solution was never found by the genetic solver, its inclusion allowed the tabu search solvers to find better solutions, more diversity among solutions in memory translating into a more effective diversification of the global search.

Several studies, including [21], showed that it is beneficial not only to include solvers of different types in the cooperation, but also to use the elite population built by these solvers in memory to construct an approximate image of the status of the

global search, e.g., to learn about the parts of the search space already explored, the relations between the values of certain decision variables (e.g., arcs in a VRP or design solution) and the value of the corresponding solution, the performance of the cooperating solvers on the particular instance given the information they receive from the central memory, etc. This information may then be used to create new knowledge, new and diverse solutions, solution components, "ideal" target solutions, etc., and guide the search. Population-based metaheuristics are particularly appropriate to generate solutions that add quality and diversity to an elite set.

Cooperative strategies including mechanisms to create new information and solutions based on the solutions exchanged belong to the p-control knowledge collegial (pC/KC) class. Most contributions to this field have solvers work on the complete problem and make the bulk of the section. We conclude the pC/KC section with a discussion on recent developments targeting multi-attribute problem settings where the problem at hand is decomposed and solvers work on particular parts of the problem or on integrating the resulting partial solutions into complete ones.

Historically, two main classes of pC/KC cooperative mechanisms are found in the literature, both based on the idea of exploiting a set of elite solutions, and their attributes, exchanged by cooperating solvers working on the complete problem, but differing in the information kept in memory. *Adaptive-memory* methods [85] 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 neighborhood and population-based metaheuristics and use them to create new solutions and knowledge to guide the cooperating solvers [18, 25, 28]. The latter method generalizes the former and, the vocabulary used in the various papers not withstanding, the two approaches are becoming increasingly unified.

The adaptive-memory terminology was coined in [85] proposing tabu search-based heuristics for the VRP and the VRPTW that are still among the most effective ones for both problems (see [3, 55, 93] for more on adaptive-memory concepts). 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 its frequency of inclusion in the best solutions encountered so far, as well as its score, and rank among the population in memory, computed from the attribute values, in particular the objective value, of its respective solutions. Solvers construct solutions out of probabilistically selected (biased by rank) solution components in memory, enhance it (tabu search 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 (routes) from different elite solutions, thus inducing a diversification effect. A number of early developments provided insights into algorithmic design. Worth mentioning are [87] for the VRPTW, where a set-covering heuristic is proposed to select the solution components in memory used to generate the new initial solution of a cooperating solver, and [51], for real-time vehicle routing and dispatching, actually implementing a hierarchical, two-level parallel scheme: a pC/KC/MPSS

cooperating adaptive memory metaheuristic at the first level, while each individual tabu-search solver implemented the route decomposition of [91] with the help of several slave processors on the second level.

Generalizing the pC/C and adaptive-memory strategies, pC/KC central-memory mechanisms keep full solutions, as well as attributes and context information sent by the solvers involved in cooperation. Solvers, which indirectly exchange complete elite solutions and context information though the central memory, may perform constructive, improving and post-optimization heuristics [64, 66, 67], neighborhood-based methods like tabu search [40, 62–64], population-based methods like genetic algorithms [40, 64, 66, 67] and path relinking [31], as well as exact solution methods [58] 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.

Other than the information received from the cooperating solvers, the central memory keeps newly created information out of these exchanged data. Statistics-building, information-extraction and learning, and new solution-creation mechanisms provide this new "knowledge". Memories recording the performance of individual solutions, solution components, and solvers may be added to the central memory, and guidance mechanisms based on this knowledge may be gradually built.

Central-memory mechanisms thus perform two main tasks: data-warehousing and communications with solvers, on the one hand, and 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/C strategies of the previous section, where solutions in memory were ordered and rank-biased randomly extracted to answer solver requests. The functions of the SC in pC/KC methods include creating new solutions, extracting appropriate solution elements, building statistics on the presence and performance of solutions, solution elements, and solvers (these belong to the family of memories, well-known in the metaheuristic community), creating the information to return when answering solver requests, the latter being part of the so-called *guidance mechanisms*.

The cooperative metaheuristic proposed in [66] for the VRPTW used a simple pC/KC mechanism, involving four solvers, two simple genetic algorithms with order and edge recombination crossovers, respectively, and two tabu search methods that perform well sequentially, Unified Tabu Search [17] and TABUROUTE [50]. The cooperating solvers shared their respective best solutions identified so far. The SC in central memory performed post-optimization (2-opt, 3-opt, Or-opt, and ejection-chain procedures to reduce the number of vehicles and the total traveled distance) on the received solutions before making them available for sharing. Solvers requested solutions from the central memory when needed, i.e., the genetic algorithms for crossover operations, the Unified Tabu at regular intervals, and TABUROUTE at diversification time. This algorithm, without any calibration or tailoring, proved to be competitive with the best metaheuristics of its day in linear speedups.

A SC enhanced with an innovative learning and guidance mechanism was proposed in [67]. The authors aimed for a mechanism that, not only returned meaningful information to solvers, but was also independent of particular problem characteristics, e.g., routes in their VRPTW application, and could be broadly applied to network-based problem settings. The SC mechanism is thus based on an atomic element in network optimization, the arc. Starting from the classical memory concepts pioneered for tabu search [53, 54, 56], 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, and, second, that this worthiness increases when the behavior appear stable in time. The authors thus considered 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 were then defined representing subsets of arcs (not necessarily adjacent) with similar frequencies of inclusion in particular population groups. Guidance was obtained by transmitting arc patterns to the individual solvers indicating whether the arcs in the pattern should be "fixed" or "prohibited" to intensify or diversify the search, respectively. The solvers accounted 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 [65] for a dynamic version of this mechanism).

The pC/KC/MPDS method proposed in [58] for the VRP illustrates how specialized solvers may address different issues in a cooperative metaheuristic, including the generation of new knowledge. Two types of solvers were defined. The so-called heuristic solvers improved solutions received from the SC associated with the central memory (called master in [58]), through a record-to-record metaheuristic [15, 57, 69]. On completing the task, the solvers returned the 50 best solutions found and the corresponding routes (a post-optimization procedure was first run on each route). Simultaneously, exact solvers aimed to identify new solutions by solving series of set covering problems starting from a limited set of routes. Each time a set covering problem was solved, the solution was returned to the central memory and the set of the current 10 best solutions was 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 was 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).

A different SC mechanism for a pC/KC metaheuristic with tabu search solvers was proposed in [63] for the VRP. Data sharing was 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 so-

lution back from it, the search being resumed from the received solution. The SC mechanism aimed to identify and extract information from the solutions in memory to guide solvers toward intensification and diversification phases. This was 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 cluster of edges and solutions being assumed to represent a region of the search space. Search history indicators were associated with clusters giving the number of solutions in the cluster and the quality of the solutions. This information was used to infer how thoroughly the corresponding region had been explored and how promising it appeared. Clusters were 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, was selected for intensification, while the solution with the lowest number of good solutions was selected for diversification. A solution was then selected in the corresponding cluster and it was 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.

We complete this section by addressing recent developments targeting multi-attribute, "rich", problem settings where the problems at hand display a large number of attributes characterizing their feasibility and optimality structures. Traditionally, such problems were simplified, or sequentially solved through a series of particular cases, where part of the overall problem was fixed or ignored, or both. The general idea of the new generation of pC/KC metaheuristics is to decompose the problem formulation along sets of decision variables, which is called *decision-set attribute decomposition* in [64]. 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. 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, the authors in [31] (see also [40]) were the first to propose such a methodology in the context of designing wireless networks, where seven attributes were considered simultaneously. The proposed pC/KC/MPDS metaheuristic had tabu search solvers working on limited subsets of attributes, the others being fixed, and a genetic method combining the partial solutions generated by the tabu search procedures into complete solutions to the initial problem.

The general method, called *Integrative Cooperative Search ICS*), was introduced in [64] (see [33, 34] for initial developments) and illustrated through an application to the multi-depot periodic vehicle routing problem (MDPVRP) [75, 108]. The main components of ICS, to be instantiated for each application, are (1) the decomposition rule; (2) the *Partial Solver Groups* (*PSGs*) addressing the partial problems resulting from the decomposition; (3) the *Integrators* selecting partial solutions from PSGs, combining them, and sending the resulting complete solutions to the *Com-*

plete Solver Group (*CSG*); and (4) the CSG, providing the central memory functionalities of ICS. 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.

The selection of the decision-sets for decomposition is specific to each application case, decision variables being clustered to yield known or identifiable optimization problem settings. Thus, an opportunistic rule decomposed the 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 PSG may contain one or several solvers targeting particular subsets of attributes. Thus, two PSGs were defined in [64], one for the PVRP and the other for the MDVRP. Each PSG was organized according to the pC/KC paradigm and was thus composed of a set of *Partial Solvers*, a central memory where elite solutions were kept, and a *Local Search Coordinator* (*LSC*) managing the local central memory and interfacing with the *Global Search Coordinator*. Two algorithms were used as partial solvers, the hybrid genetic algorithm HGSADC [108] and GUTS, a generalized version of the Unified Tabu Search [17].

Integrators build complete solutions by mixing partial solutions with promising features obtained within the PSGs. Integrators aim for solution quality, the transmission of critical features extracted from the partial solutions, and computational efficiency. 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 metaheuristics 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 [43] belongs to the latter category, proposing particular optimization models for rich VRP settings, which preserve desired critical variables (desired attributes), present in partial solutions, when selecting and combining routes.

Several Integrators can be involved in an ICS metaheuristic, 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 crossover and individual education (enhancement) operators of HGSADC, and two of the methods proposed in [43], 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 started from pairs of partial solutions randomly selected among the best 25% of the solutions in the central memories of the two PSGs.

The Complete Solver Group includes the central memory, where the complete solutions are stored, together with the context information and the guiding solutions built by the Global Search Coordinator (GSC). Complete solutions are received from Integrators and, when solvers are present in the CSG, these solutions are further enhanced. The GSC (1) builds the contextual information (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), (2) generates new guiding solutions to orient the search toward promising features, and (3) monitors the status of the solver groups, sending guiding instructions (solutions) when necessary.

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 solution 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. In [64], guidance took the form of three solutions, which were either randomly selected from the complete solution set, or were built by the GSC out of promising solution elements with respect to the search history.

The authors in [64] reported very good results even when compared to the state-of-the-art metaheuristic. 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) when using genetic solvers in a PSG it is preferable for long runs to define a local population for each such solver, while using the central memory as population for all cooperating genetic solvers appears better for short runs; and (3) embedding good solvers in the CSG enhances slightly the already excellent performance of the ICS parallel metaheuristic.

13.8 Conclusions

This chapter presented an overview and state-of-the-art survey of the main parallel metaheuristic ideas, discussing general concepts and algorithm design principles and strategies. Four main classes of parallel metaheuristics strategies were described: low-level decomposition of computing-intensive tasks with no modification to the original algorithm, decomposition of the search space, independent multi-search, and cooperative multi-search, the latter encompassing synchronous, asynchronous collegial and knowledge-creating asynchronous collegial strategies. It is noteworthy that this series also reflects the historical sequence of the development of parallel metaheuristics, which are now acknowledged, cooperative search strategies in particular, as making up their own class of metaheuristics.

It must be emphasized that each of these strategy classes fulfills a particular type of task and all are needed at some time. 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 neighborhood in classic metaheuristic frameworks but, as a number of more 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 metaheuristics 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 behavior and identify the most appropriate combination of strategies, particularly low-level and cooperative search, for various metaheuristics, 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. Let us not forget, however, that most optimization problems of interest are complex and that the dimensions of the instances one faces in practice keep increasing. Research challenges exist in dynamic search-space decomposition and the combination of cooperative search and search-space decomposition. The Integrative Cooperative Search is a first answer in this direction, but more research is needed.

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

A first issue concerns the exchange and utilization of context data locally generated by the cooperating solvers, to infer an image of the status of the global search and generate appropriate guiding instructions. Thus, contrasting the various local context data 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 metaheuristics, as well as the cooperation of metaheuristics with exact solution methods. The so-called hybrid and matheuristic methods, representing the former and latter types of method combination, respectively, are trendy in the sequential optimization field. Very few studies explicitly target parallel methods, however. 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 metaheuristic 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. Transversal studies comparing the behavior and performance of particular parallel metaheuristic 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 metaheuristics.

Acknowledgements The author wishes to acknowledge the contributions of colleagues and students, in particular Professors Michel Gendreau, Université de Montréal, Canada, and Michel Toulouse, the Vietnamese-German University, Vietnam, who collaborated over the years to the work on parallel metaheuristics for combinatorial optimization. All errors are, however, solely and entirely due to the author.

Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Discovery Grant and the Discovery Accelerator Supplements programs, and the Strategic Clusters program of the Fonds de Recherche Québécois Nature et Technologies (FRQNT). The author thanks the two institutions for supporting this research.

References

1. E. Alba (ed.), *Parallel Metaheuristics: A New Class of Algorithms* (Wiley, Hoboken, 2005)
2. E. Alba, G. Luque, S. Nesmachnow, Parallel metaheuristics: recent advances and new trends. Int. Trans. Oper. Res. **20**(1), 1–48 (2013)
3. P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, É.D. Taillard, A parallel tabu search heuristic for the vehicle routing problem with time windows. Transp. Res. C: Emerg. Technol. **5**(2), 109–122 (1997)
4. R. Banos, J. Ortega, C. Gil, A. Fernandez, F. de Toro, A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. Exp. Syst. Appl. **40**(5), 1696–1707 (2013)
5. R.S. Barr, B.L. Hickman, Reporting computational experiments with parallel algorithms: issues, measures, and experts opinions. ORSA J. Comput. **5**(1), 2–18 (1993)
6. M.P. Bastos, C.C. Ribeiro, Reactive tabu search with path-relinking for the steiner problem in graphs, in *Meta-Heuristics 98: Theory & Applications*, ed. by S. Voß, S. Martello, C. Roucairol, I.H. Osman (Kluwer Academic Publishers, Norwell, 1999), pp. 31–36
7. R. Battiti, G. Tecchiolli, Parallel based search for combinatorial optimization: genetic algorithms and TABU. Microprocess. Microsyst. **16**(7), 351–367 (1992)
8. J. Blazewicz, A. Moret-Salvador, R. Walkowiak, Parallel tabu search approaches for two-dimensional cutting. Parallel Process. Lett. **14**(1), 23–32 (2004)
9. A. Bortfeldt, H. Gehring, D. Mack, A parallel tabu search algorithm for solving the container loading problem. Parallel Comput. **29**(5), 641–662 (2003)
10. A.R. Brodtkorb, T.R. Hagen, C. Schulz, G. Hasle, GPU computing in discrete optimization. Part I: introduction to the GPU. EURO J. Transp. Logist. **2**(1–2), 129–157 (2013)
11. A.R. Brodtkorb, T.R. Hagen, C. Schulz, G. Hasle, GPU computing in discrete optimization. Part II: survey focussed on routing problems. EURO J. Transp. Logist. **2**(1–2), 159–186 (2013)

12. E. Cantú-Paz, Theory of parallel genetic algorithms, in *Parallel Metaheuristics: A New Class of Algorithms*, ed. by E. Alba (Wiley, Hoboken, 2005), pp. 425–445
13. C.B.C. Cavalcante, V.F. Cavalcante, C.C. Ribeiro, M.C. Souza, Parallel cooperative approaches for the labor constrained scheduling problem, in *Essays and Surveys in Metaheuristics*, ed. by C.C. Ribeiro, P. Hansen (Kluwer Academic Publishers, Norwell, 2002), pp. 201–225
14. J.M. Cecilia, J.M. Garciá, A. Nisbet, M. Amos, M. Ujaldón, Enhancing data parallelism for ant colony optimization on GPUs. J. Parallel Distrib. Comput. **73**(1), 42–51 (2013)
15. I.M. Chao, B.L. Golden, E.A. Wasil, An improved heuristic for the period vehicle routing problem. Networks **26**(1), 25–44 (1995)
16. J.-F. Cordeau, M. Maischberger, A parallel iterated tabu search heuristic for vehicle routing problems. Comput. Oper. Res. **39**(9), 2033–2050 (2012)
17. J.-F. Cordeau, G. Laporte, A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. **52**(8), 928–936 (2001)
18. T.G. Crainic, Parallel computation, co-operation, tabu search, in *Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search*, ed. by C. Rego, B. Alidaee (Kluwer Academic Publishers, Norwell, 2005), pp. 283–302
19. T.G. Crainic, Parallel solution methods for vehicle routing problems, in *The Vehicle Routing Problem: Latest Advances and New Challenges*, ed. by B.L. Golden, S. Raghavan, E.A. Wasil (Springer, New York, 2008), pp. 171–198
20. T.G. Crainic, Parallel meta-heuristic search, in *Handbook of Heuristics*, ed. by R. Marti, P.M. Pardalos, M.G.C. Resende (Springer, New York, 2017)
21. T.G. Crainic, M. Gendreau, Towards an evolutionary method - cooperating multi-thread parallel tabu search hybrid, in *Meta-Heuristics 98: Theory & Applications*, ed. by S. Voß, S. Martello, C. Roucairol, I.H. Osman (Kluwer Academic Publishers, Norwell, 1999), pp. 331–344
22. T.G. Crainic, M. Gendreau, Cooperative parallel tabu search for capacitated network design. J. Heuristics **8**(6), 601–627 (2002)
23. T.G. Crainic, N. Hail, Parallel meta-heuristics applications, in *Parallel Metaheuristics: A New Class of Algorithms*, ed. by E. Alba (Wiley, Hoboken, 2005), pp. 447–494
24. T.G. Crainic, M. Toulouse, Parallel metaheuristics, in *Fleet Management and Logistics*, ed. by T.G. Crainic, G. Laporte (Kluwer Academic Publishers, Norwell, 1998), pp. 205–251
25. T.G. Crainic, M. Toulouse, Parallel strategies for meta-heuristics, in *Handbook in Metaheuristics*, ed. by F. Glover, G. Kochenberger (Kluwer Academic Publishers, Norwell, 2003), pp. 475–513
26. T.G. Crainic, M. Toulouse, Explicit and emergent cooperation schemes for search algorithms, in *Learning and Intelligent Optimization*, ed. by V. Maniezzo, R. Battiti, J.-P. Watson. Lecture Notes in Computer Science, vol. 5315 (Springer, Berlin, 2008), pp. 95–109
27. T.G. Crainic, M. Toulouse, Parallel meta-heuristics, in *Handbook of Metaheuristics*, ed. by M. Gendreau, J.-Y. Potvin, 2nd edn. (Springer, Berlin, 2010), pp. 497–541
28. T.G. Crainic, M. Toulouse, M. Gendreau, Parallel asynchronous tabu search for multicommodity location-allocation with balancing requirements. Ann. Oper. Res. **63**, 277–299 (1996)
29. T.G. Crainic, M. Toulouse, M. Gendreau, Towards a taxonomy of parallel tabu search algorithms. INFORMS J. Comput. **9**(1), 61–72 (1997)
30. T.G. Crainic, M. Gendreau, P. Hansen, N. Mladenović, Cooperative parallel variable neighborhood search for the *p*-median. J. Heuristics **10**(3), 293–314 (2004)
31. T.G. Crainic, B. Di Chiara, M. Nonato, L. Tarricone, Tackling electrosmog in completely configured 3G networks by parallel cooperative meta-heuristics. IEEE Wireless Commun. **13**(6), 34–41 (2006)
32. T.G. Crainic, Y. Li, M. Toulouse, A first multilevel cooperative algorithm for the capacitated multicommodity network design. Comput. Oper. Res. **33**(9), 2602–2622 (2006)
33. T.G. Crainic, G.C. Crisan, M. Gendreau, N. Lahrichi, W. Rei, A concurrent evolutionary approach for cooperative rich combinatorial optimization, in *Genetic and Evolutionary Computation Conference - GECCO 2009*, July 8–12, Montréal, Canada (ACM, New York, 2009). CD-ROM

34. T.G. Crainic, G.C. Crisan, M. Gendreau, N. Lahrichi, W. Rei, Multi-thread integrative cooperative optimization for rich combinatorial problems, in *The 12th International Workshop on Nature Inspired Distributed Computing - IDISC'09*, 25–29 May, Rome (2009). CD-ROM
35. T.G. Crainic, T. Davidović, D. Ramljak, Designing parallel meta-heuristic methods, in *High Performance and Cloud Computing in Scientific Research and Education*, ed. by M. Despotovic-Zrakic, V. Milutinovic, A. Belic (IGI Global, Hershey, 2014), pp. 260–280
36. Z.J. Czech, A parallel genetic algorithm for the set partitioning problem, in *8th Euromicro Workshop on Parallel and Distributed Processing* (2000), pp. 343–350
37. C. Dai, B. Li, M. Toulouse, A multilevel cooperative tabu search algorithm for the covering design problem. J. Comb. Math. Comb. Comput. **68**, 35–65 (2009)
38. T. Davidović, T.G. Crainic, Parallel local search to schedule communicating tasks on identical processors. Parallel Comput. **48**, 1–14 (2015)
39. A. Delévacq, P. Delisle, M. Gravel, M. Krajecki, Parallel ant colony optimization on graphics processing units. J. Parallel Distrib. Comput. **73**(1), 52–61 (2013)
40. B. Di Chiara, Optimum planning of 3G cellular systems: radio propagation models and cooperative parallel meta-heuristics. Ph.D. thesis, Dipartimento di ingegneria dell'innovatione, Universitá degli Studi di Lecce, Lecce, 2006
41. K.F. Doerner, R.F. Hartl, S. Benkner, M. Lucka, Cooperative savings based ant colony optimization - multiple search and decomposition approaches. Parallel Process. Lett. **16**(3), 351–369 (2006)
42. H. Drias, A. Ibri, Parallel ACS for weighted MAX-SAT, in *Artificial Neural Nets Problem Solving Methods - Proceedings of the 7th International Work-Conference on Artificial and Natural Neural Networks*, ed. by J. Mira, J. Álvarez. Lecture Notes in Computer Science, vol. 2686 (Springer, Heidelberg, 2003), pp. 414–421
43. N. El Hachemi, T.G. Crainic, N. Lahrichi, W. Rei, T. Vidal, Solution integration in combinatorial optimization with applications to cooperative search and rich vehicle routing. J. Heuristics **21**(5), 663–685 (2015)
44. C.D. Flores, B.B. Cegla, D.B. Caceres, Telecommunication network design with parallel multi-objective evolutionary algorithms, in *Proceedings of the 2003 IFIP/ACM Latin America Networking Conference - Towards a Latin American Agenda for Network Research* (ACM, New York, 2003), pp. 1–11
45. F. García-López, B. Melián-Batista, J.A. Moreno-Pérez, J.M. Moreno-Vega, The parallel variable neighborhood search for the *p*-median problem. J. Heuristics **8**(3), 375–388 (2002)
46. F. García-López, B. Melián-Batista, J.A. Moreno-Pérez, J.M. Moreno-Vega, Parallelization of the scatter search for the *p*-median problem. Parallel Comput. **29**, 575–589 (2003)
47. F. García-López, M. García Torres, B. Melián-Batista, J.A. Moreno-Pérez, J.M. Moreno-Vega, Parallel scatter search, in *Parallel Metaheuristics: A New Class of Metaheuristics* (Wiley, Hoboken, 2005), pp. 223–246
48. F. García-López, M. García Torres, B. Melián-Batista, J.A. Moreno-Pérez, J.M. Moreno-Vega, Solving feature subset selection problem by a parallel scatter search. Eur. J. Oper. Res. **169**(2), 477–489 (2006)
49. H. Gehring, J. Homberger, Parallelization of a two-phase metaheuristic for routing problems with time windows. J. Heuristics **8**(3), 251–276 (2002)
50. M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem. Manag. Sci. **40**(10), 1276–1290 (1994)
51. M. Gendreau, F. Guertin, J.-Y. Potvin, É.D. Taillard, Tabu search for real-time vehicle routing and dispatching. Transp. Sci. **33**(4), 381–390 (1999)
52. M. Gendreau, G. Laporte, F. Semet, A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. Parallel Comput. **27**(12), 1641–1653 (2001)
53. F. Glover, Tabu search – Part I. ORSA J. Comput. **1**(3), 190–206 (1989)
54. F. Glover, Tabu search – Part II. ORSA J. Comput. **2**(1), 4–32 (1990)
55. F. Glover, Tabu search and adaptive memory programming – advances, applications and challenges, in *Interfaces in Computer Science and Operations Research*, ed. by R.S. Barr, R.V. Helgason, J. Kennington (Kluwer Academic Publishers, Norwell, 1996), pp. 1–75

56. F. Glover, M. Laguna, *Tabu Search* (Kluwer Academic Publishers, Norwell, 1997)
57. B.L. Golden, E.A. Wasil, J.P. Kelly, I.M. Chao, Metaheuristics in vehicle routing, in *Fleet Management and Logistics*, ed. by T.G. Crainic, G. Laporte (Kluwer Academic Publishers, Norwell, 1998), pp. 33–56
58. C. Groër, B. Golden, A parallel algorithm for the vehicle routing problem. INFORMS J. Comput. **23**(2), 315–330 (2011)
59. J.I. Hidalgo, M. Prieto, J. Lanchares, R. Baraglia, F. Tirado, O. Garnica, Hybrid parallelization of a compact genetic algorithm, in *Proceedings of the 11th Uromicro Conference on Parallel, Distributed and Network-Based Processing* (2003), pp. 449–455
60. D. Izzo, M. Rucinski, C. Ampatzis, Parallel global optimisation meta-heuristics using an asynchronous island-model, in *CEC'09 - IEEE Congress on Evolutionary Computation* (IEEE, Piscataway, 2009), pp. 2301–2308
61. T. James, C. Rego, F. Glover, A cooperative parallel tabu search algorithm for the quadratic assignment problem. Eur. J. Oper. Res. **195**(3), 810–826 (2009)
62. J. Jin, T.G. Crainic, A. Løkketangen, A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. Eur. J. Oper. Res. **222**(3), 441–451 (2012)
63. J. Jin, T.G. Crainic, A. Løkketangen, A cooperative parallel metaheuristic for the capacitated vehicle routing problem. Comput. Oper. Res. **44**, 33–41 (2014)
64. N. Lahrichi, T.G. Crainic, M. Gendreau, W. Rei, C.C. Crisan, T. Vidal, An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. Eur. J. Oper. Res. **246**(2), 400–412 (2015)
65. A. Le Bouthillier, Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire. Ph.D. thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, 2007
66. A. Le Bouthillier, T.G. Crainic, A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Comput. Oper. Res. **32**(7), 1685–1708 (2005)
67. A. Le Bouthillier, T.G. Crainic, P. Kropf, A guided cooperative search for the vehicle routing problem with time windows. IEEE Intell. Syst. **20**(4), 36–42 (2005)
68. S.Y. Lee, K.G. Lee, Synchronous and asynchronous parallel simulated annealing with multiple Markov chains. IEEE Trans. Parallel Distrib. Syst. **7**(10), 993–1007 (1996)
69. F. Li, B.L. Golden, E.A. Wasil, Very large-scale vehicle routing: new test problems, algorithms, and results. Comput. Oper. Res. **32**(5), 1165–1179 (2005)
70. C. Ling, S. Hai-Ying, W. Shu, A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem. Inf. Sci. **199**, 31–42 (2012)
71. N. Melab, E.-G. Talbi, S. Cahon, E. Alba, G. Luque, Parallel metaheuristics: models and frameworks, in *Parallel Combinatorial Optimization*, ed. by E.-G. Talbi (Wiley, New York, 2006), pp. 149–162
72. N. Melab, T.-V. Luong, K. Boufaras, E.-G. Talbi, Towards paradisEO-MO-GPU: a framework for gpu-based local search metaheuristics, in *Advances in Computational Intelligence*. Lecture Notes in Computer Science, vol. 6691, ed. by J. Cabestany, I. Rojas, G. Joya (Springer, Berlin, 2011), pp. 401–408
73. R. Michels, M. Middendorf, An ant system for the shortest common supersequence problem, in *New Ideas in Optimization*, ed. by D. Corne, M. Dorigo, F. Glover (McGraw-Hill, New York, 1999), pp. 51–61
74. M. Middendorf, F. Reischle, H. Schmeck, Multi colony ant algorithms. J. Heuristics **8**(3), 305–320 (2002)
75. A. Mingozzi, The multi-depot periodic vehicle routing problem, in *Abstraction, Reformulation and Approximation*, ed. by J.-D. Zucker, L. Saitta. Lecture Notes in Computer Science, vol. 3607 (Springer, Berlin, 2005), pp. 347–350
76. S. Niar, A. Fréville, A parallel tabu search algorithm for the 0–1 multidimensional knapsack problem, in *11th International Parallel Processing Symposium (IPPS '97)*, Geneva (IEEE, Piscataway, 1997), pp. 512–516

77. I.O. Oduntan, M. Toulouse, R. Baumgartner, C. Bowman, R. Somorjai, T.G. Crainic, A multilevel tabu search algorithm for the feature selection problem in biomedical data sets. Comput. Math. Appl. **55**(5), 1019–1033 (2008)
78. M. Ouyang, M. Toulouse, K. Thulasiraman, F. Glover, J.S. Deogun, Multi-level cooperative search: application to the netlist/hypergraph partitioning problem, in *Proceedings of International Symposium on Physical Design* (ACM Press, New York, 2000), pp. 192–198
79. M. Ouyang, M. Toulouse, K. Thulasiraman, F. Glover, J.S. Deogun, Multilevel cooperative search for the circuit/hypergraph partitioning problem. IEEE Trans. Comput. Aided Des. **21**(6), 685–693 (2002)
80. M. Pedemonte, S. Nesmachnow, H. Cancela, A survey of parallel ant colony optimization. Appl. Soft Comput. **11**(8), 5181–5197 (2011)
81. M. Polacek, S. Benkner, K.F. Doerner, R.F. Hartl, A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. Bus. Res. **1**(2), 207–218 (2008)
82. C. Rego, Node ejection chains for the vehicle routing problem: sequential and parallel algorithms. Parallel Comput. **27**(3), 201–222 (2001)
83. C.C. Ribeiro, I. Rosseti, Efficient parallel cooperative implementations of GRASP heuristics. Parallel Comput. **33**(1), 21–35 (2007)
84. E. Rios, L.S. Ochi, C. Bœres, V.N. Cœlho, I.M. Cœlho, R. Faria, Exploring parallel multi-GPU local search strategies in a metaheuristic framework. J. Parallel Distrib. Comput. (2017). https://doi.org/10.1016/j.jpdc.2017.06.011
85. Y. Rochat, É.D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing. J. Heuristics **1**(1), 147–167 (1995)
86. H. Sanvicente-Sánchez, J. Frausto-Solís, MPSA: a methodology to parallelize simulated annealing and its application to the traveling salesman problem, in *MICAI 2002: Advances in Artificial Intelligence*, ed. by C.A. Coello Coello, A. de Albornoz, L.E. Sucar, O.C. Battistutti. Lecture Notes in Computer Science, vol. 2313 (Springer, Heidelberg, 2002), pp. 89–97
87. J. Schulze, T. Fahle, A parallel algorithm for the vehicle routing problem with time window constraints. Ann. Oper. Res. **86**, 585–607 (1999)
88. M. Sevkli, M.E. Aydin, Parallel variable neighbourhood search algorithms for job shop scheduling problems. IMA J. Manag. Math. **18**(2), 117–133 (2007)
89. M. Solar, V. Parada, R. Urrutia, A parallel genetic algorithm to solve the set-covering problem. Comput. Oper. Res. **29**(9), 1221–1235 (2002)
90. T. Stutzle, Parallelization strategies for ant colony optimization, in *Proceedings of Parallel Problem Solving from Nature V*, ed. by A.E. Eiben, T. Back, M. Schoenauer, H.-P. Schwefel. Lecture Notes in Computer Science, vol. 1498 (Springer, Heidelberg, 1998), pp. 722–731
91. É.D. Taillard, Parallel iterative search methods for vehicle routing problems. Networks **23**(8), 661–673 (1993)
92. É.D. Taillard, Parallel taboo search techniques for the job shop scheduling problem. ORSA J. Comput. **6**(2), 108–117 (1994)
93. É.D. Taillard, L.M. Gambardella, M. Gendreau, J.-Y. Potvin, Adaptive memory programming: a unified view of metaheuristics. Eur. J. Oper. Res. **135**(1), 1–10 (1997)
94. E.-G. Talbi (ed.), *Parallel Combinatorial Optimization* (Wiley, Hoboken, 2006)
95. E.-G. Talbi (ed.), *Metaheuristics: From Design to Implementation* (Wiley, Hoboken, 2009)
96. S. Talukdar, S. Murthy, R. Akkiraju, Asynchronous teams, in *Handbook in Metaheuristics*, ed. by F. Glover, G. Kochenberger (Kluwer Academic Publishers, Norwell, 2003)
97. Y. Tan, K. Ding, A survey on GPU-based implementation of swarm intelligence algorithms. IEEE Trans. Cybern. **46**(9), 2168–2267 (2016)
98. H.M.M. ten Eikelder, B.J.L. Aarts, M.G.A. Verhoeven, E.H.L. Aarts, Sequential and parallel local search for job shop scheduling, in *Meta-Heuristics 98: Theory & Applications*, ed. by S. Voß, S. Martello, C. Roucairol, I.H. Osman (Kluwer Academic Publishers, Norwell, 1999), pp. 359–371
99. G. Tongcheng, M. Chundi, 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 (2002), pp. 1840–1843

100. M. Toulouse, T.G. Crainic, M. Gendreau, Communication issues in designing cooperative multi thread parallel searches, in *Meta-Heuristics: Theory & Applications*, ed. by I.H. Osman, J.P. Kelly (Kluwer Academic Publishers, Norwell, 1996), pp. 501–522
101. M. Toulouse, T.G. Crainic, B. Sansó, K. Thulasiraman, Self-organization in cooperative search algorithms, in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics* (Omnipress, Madison, 1998), pp. 2379–2385
102. M. Toulouse, T.G. Crainic, B. Sansó, An experimental study of systemic behavior of cooperative search algorithms, in *Meta-Heuristics 98: Theory & Applications*, ed. by S. Voß, S. Martello, C. Roucairol, I.H. Osman (Kluwer Academic Publishers, Norwell, 1999), pp. 373–392
103. M. Toulouse, K. Thulasiraman, F. Glover, Multi-level cooperative search: a new paradigm for combinatorial optimization and an application to graph partitioning, in *5th International Euro-Par Parallel Processing Conference*, ed. by P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, D. Ruiz. Lecture Notes in Computer Science, vol. 1685 (Springer, Heidelberg, 1999), pp. 533–542
104. M. Toulouse, T.G. Crainic, K. Thulasiraman, Global optimization properties of parallel cooperative search algorithms: a simulation study. Parallel Comput. **26**(1), 91–112 (2000)
105. M. Toulouse, T.G. Crainic, B. Sansó, Systemic behavior of cooperative search algorithms. Parallel Comput. **30**(1), 57–79 (2004)
106. E. Vallada, R. Ruiz, A cooperative metaheuristics for the permutation flowshop scheduling problem. Eur. J. Oper. Res. **193**(2), 365–376 (2009)
107. T. Van Luong, N. Melab, E.-G. Talbi, GPU computing for parallel local search metaheuristic algorithms. IEEE Trans. Comput. **62**(1), 173–185 (2013)
108. T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)