(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2018 (29.11.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/217992 A1

(51) International Patent Classification:
G06Q 10/06 (2012.01)

(21) International Application Number:
PCT/US2018/034338

(22) International Filing Date:
24 May 2018 (24.05.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/606,084 26 May 2017 (26.05.2017) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, CA 94065 (US).

(72) Inventors: **VAN'T WESTEINDE, Charles, P.**; 13 Stanley St., Black Rock, Victoria, 3193 (AU). **MIZINA, Svetlana**; 118 Clarinda Rd., Clarinda, Victoria, Melbourne, 3193 (AU). **ZOUBTCHENKO, Marina**; 13 Stanley St., Black Rock, Victoria, 3193 (AU).

(74) Agent: **KRAGULJAC, Petar**; KRAGULJAC LAW GROUP, LLC, 4700 Rockside Road, Summit One - Suite 510, Independence, OH 44131 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: COMPUTERIZED SYSTEM AND METHOD FOR RESOLVING CROSS-VEHICLE DEPENDENCIES FOR VEHICLE SCHEDULING

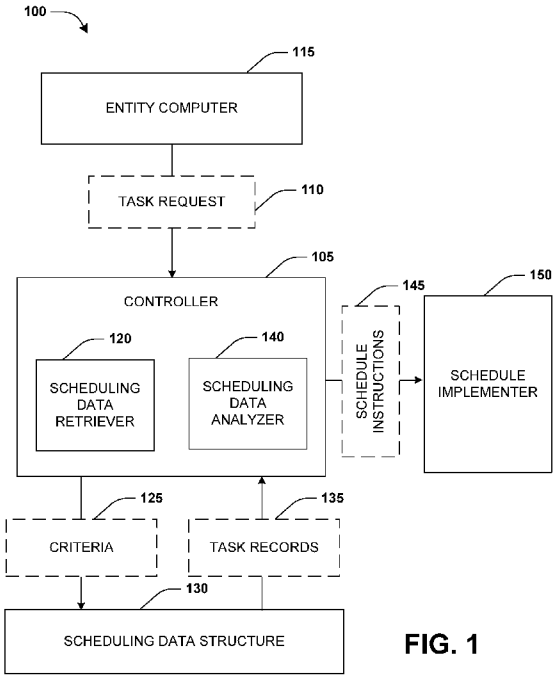

FIG. 1

(57) Abstract: Systems, methods, and other embodiments associated with computing and assigning a task to be performed using a resource at a candidate time are described. In one embodiment, a method includes accessing a scheduling data structure and analyzing data records for tasks, upon which the resource depends, to be performed using one or more other resources, and tasks, upon which other resources depend, to be performed using the resource. A candidate time for performing the task using the resource is calculated based upon dependencies determined between various tasks, and a candidate schedule is generated using the candidate time for the task. The scheduling data structure is modified and regenerated based upon the candidate schedule if the candidate schedule is determined to have a greater performance score than an existing schedule.

WO 2018/217992 A1

COMPUTERIZED SYSTEM AND METHOD FOR RESOLVING CROSS-VEHICLE DEPENDENCIES FOR VEHICLE SCHEDULING

BACKGROUND

[0001] Computing devices are used to implement various services and products. A computing device may provide a scheduling service, such as scheduling of tasks to be performed using vehicles. Scheduled tasks may be stored within databases or other storage structures of a distributed network environment (e.g., a cloud service), or within databases or other storage structures of a local computer. A user may interact with the computing device to perform operations upon the scheduled tasks. For example, the computing device may be used to add new tasks, to remove existing tasks, or to modify existing tasks.

[0002] Various tasks managed by the scheduling service may be scheduled to be performed using various vehicles, such that some tasks may be scheduled to be performed using a first vehicle, and other tasks may be scheduled to be performed using a second vehicle. There may be a dependency between a first task scheduled to be performed using the first vehicle and a second task scheduled to be performed using the second vehicle, which may complicate attempts to modify an existing schedule.

[0003] In order to improve the effectiveness of vehicles, the rate of successful completion of tasks, increase associated profits, and/or move inventory being delivered by the vehicles, it is desirable to efficiently identify and assign times for new tasks.

[0004] Unfortunately, typical existing techniques are limited to identifying some times, and not others that could be utilized. Thus, generating schedules has been restricted by having limited choices of time assignments.

SUMMARY

[0005] In one embodiment, a computing system is disclosed that comprises: a processor connected to memory; and a scheduling module stored on a non-transitory computer readable medium and configured with instructions that when executed by the processor cause the processor to: in response to receiving a request to modify a scheduling data structure that includes a plurality of tasks, determine that the request is directed towards scheduling a first task to be performed using a first resource from a plurality of resources; and/or access the scheduling data structure from a database via a network communication, wherein the scheduling data structure includes data records for: (i) a first set of tasks, upon which the first resource depends, to be performed using one or more other resources, and (ii) a second set of tasks, upon which other resources depend, to be performed using the first resource; and/or analyze the data records to determine dependencies between the first set of tasks to be performed using the one or more other resources and the second set of tasks to be performed using the first resource; and/or calculate a candidate time for performing the first task using the first resource based upon the dependencies; and/or generate a candidate schedule for the plurality of resources, including the first resource, using the candidate time for the first task for the first resource; and/or evaluate the candidate schedule to calculate a candidate schedule performance score; and/or in response to determining that the candidate schedule performance score is greater than an existing schedule performance score of an existing schedule, modify and regenerate the scheduling data structure, based upon the candidate schedule, to assign the first task to be performed using the first resource at the candidate time.

[0006] In another embodiment of the computing system, the instructions when executed by the processor cause the processor to iteratively: calculate a second candidate time for performing a second task using a second resource, from the plurality of resources, based upon one or more dependencies; and/or generate a second candidate schedule for the plurality of resources, including the second resource, using the second candidate time for the second task for the second resource; and/or evaluate the second candidate schedule to calculate a second candidate schedule performance score; and/or in response to determining that the second candidate schedule performance score is not greater than a second existing schedule performance score of a second existing schedule, calculate a third candidate time for performing the second task using the second resource, from the plurality of resources, based upon the one or more dependencies; and/or in response to determining that the second candidate schedule performance score is greater than the second existing schedule performance score of the second existing schedule, modify and regenerate the scheduling data structure, based upon the second candidate schedule, to assign the second task to be performed using the second resource at the second candidate time.

[0007] In another embodiment of any combination of components from the computing system, the instructions to evaluate the candidate schedule to calculate the candidate schedule performance score further include instructions that when executed by the processor cause the processor to: analyze the candidate schedule to determine a number of conflicts in dependencies between the first set of tasks and the second set of tasks in the candidate schedule; and generate the candidate schedule performance score based upon the number of conflicts in dependencies.

[0008] In another embodiment of any combination of components from the computing system, the conflicts in dependencies comprise at least one of: (i) a conflict corresponding to a second task to be performed using the first resource having a time before a third task to be performed using a second resource, wherein

the dependencies comprise a dependency of the second task to be performed using the first resource on the third task to be performed using the second resource; or (ii) a conflict corresponding to a fourth task to be performed using the second resource having a time before a fifth task to be performed using the first resource, wherein the dependencies comprise a dependency of the fourth task to be performed using the second resource on the fifth task to be performed using the first resource.

[0009] In another embodiment of any combination of components from the computing system, the dependencies used to calculate the candidate time for performing the first task using the first resource comprise at least one of: (i) a dependency of the first task to be performed using the first resource on a second task to be performed using a second resource; or (ii) a dependency of a third task to be performed using the second resource on the first task to be performed using the first resource; and/or the instructions to calculate the candidate time for performing the first task using the first resource based upon the dependencies further include instructions that when executed by the processor cause the processor to at least one of: calculate the candidate time to be after a time of the second task based upon the dependency of the first task to be performed using the first resource on the second task to be performed using the second resource; or calculate the candidate time to be before a time of the third task based upon the dependency of the third task to be performed using the second resource on the first task to be performed using the first resource.

[0010] In another embodiment of any combination of components from the computing system, the dependencies used to calculate the candidate time for performing the first task using the first resource comprise at least one of: (i) a dependency of a second task for the first resource on a third task for a second resource; or (ii) a dependency of a fourth task for the second resource on a fifth task for the first resource; and/or wherein the instructions when executed by the

processor cause the processor to at least one of: calculate a second candidate time for the second task for the first resource to be after a time of the third task based upon the dependency of the second task for the first resource on the third task for the second resource; or calculate a third candidate time for the fifth task for the first resource to be before a time of the fourth task based upon the dependency of the fourth task for the second resource on the fifth task for the first resource; and/or wherein the instructions to generate the candidate schedule for the plurality of resources further include instructions that when executed by the processor cause the processor to at least one of: use the second candidate time for the second task for the first resource in the candidate schedule; or use the third candidate time for the fifth task for the first resource in the candidate schedule.

[0011] In another embodiment of any combination of components from the computing system, the first resource is a first vehicle, the first set of tasks are to be performed using one or more other vehicles, and the second set of tasks are to be performed by the first vehicle; and/or wherein the instructions to determine that the candidate schedule performance score is greater than the existing schedule performance score of the existing schedule further include instructions that when executed by the processor cause the processor to: determine that the candidate schedule resolves a conflict in the existing schedule caused by a dependency between the first vehicle and at least one of the one or more other vehicles.

[0012] In another embodiment, a computer-implemented method performed by a computing device comprising a processor is disclosed. The computer-implemented method comprising one or more combinations of functions: receiving, by at least the processor, a request to modify a scheduling data structure that includes a plurality of tasks; and/or accessing, by at least the processor, the scheduling data structure from a database via a network communication, wherein the scheduling data structure includes data records for: (i) a first set of tasks, upon which a first resource depends, to be performed using one or more other

resources, and/or (ii) a second set of tasks, upon which other resources depend, to be performed using the first resource; and/or analyzing, by at least the processor, the data records to determine dependencies between the first set of tasks to be performed using the one or more other resources and the second set of tasks to be performed using the first resource; and/or calculating, by at least the processor, a candidate time for performing a first task using the first resource based upon the dependencies; and/or generating, by at least the processor, a candidate schedule for the plurality of resources, including the first resource, using the candidate time for the first task for the first resource; and/or evaluating, by at least the processor, the candidate schedule to calculate a candidate schedule performance score; and/or in response to determining that the candidate schedule performance score is greater than an existing schedule performance score of an existing schedule, modifying and regenerating, by at least the processor, the scheduling data structure, based upon the candidate schedule, to assign the first task to be performed using the first resource at the candidate time.

[0013] In another embodiment, the method may include any combination of functions as described as part of the computing system.

[0014] In another embodiment, a non-transitory computer readable medium is disclosed that comprises stored instructions (e.g., as one or more programs or algorithms) that when executed by a processor of a computing device, perform any combination of functions of the disclosed method.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate various systems, methods, and other embodiments of the disclosure. It will be appreciated that the illustrated element boundaries (e.g., boxes, groups of boxes, or other shapes) in the figures represent

one embodiment of the boundaries. In some embodiments one element may be implemented as multiple elements or that multiple elements may be implemented as one element. In some embodiments, an element shown as an internal component of another element may be implemented as an external component and vice versa. Furthermore, elements may not be drawn to scale.

[0016] FIG. 1 illustrates an embodiment of a system associated with assigning a task to be performed using a resource at a candidate time.

[0017] FIG. 2 illustrates an embodiment of a method associated with assigning the task to be performed using the resource at the candidate time.

[0018] FIG. 3 illustrates an embodiment of the scheduling data structure.

[0019] FIG. 4 illustrates an embodiment of an entity computer and a graphical user interface.

[0020] FIG. 5 illustrates an embodiment of a schedule implementer with scheduling instructions.

[0021] FIG. 6 illustrates an embodiment of the schedule implementer with a monitor interface.

[0022] FIG. 7 illustrates an embodiment of a non-transitory computer-readable medium.

[0023] FIG. 8 illustrates an embodiment of a computing system configured with the example systems and/or methods disclosed.

DETAILED DESCRIPTION

[0024] Computerized scheduling systems and methods are described herein that provide for scheduling a task(s), such as a delivery, to be performed using a resource(s), such as a vehicle, a loading bay, a crew/team, etc. In one embodiment, an electronic schedule is a data structure configured with a number

of time slots. The electronic schedule may include different tasks that are scheduled to be performed at different times in different time slots. However, one or more tasks may have a dependency relationship with another task. For example, a first delivery by a first vehicle may be dependent upon another delivery by a second vehicle. Thus, the first delivery may need to be performed after performance/completion of the other delivery by the second vehicle. These types of dependencies affect and/or restrict how the tasks are scheduled in the electronic schedule.

[0025] When a user attempts to schedule a new task via the scheduling system, the user may provide input about the new task, such as a resource to be used to perform the new task, a dependency relationship between the new task and another task, and a deadline by which the new task needs to be completed. The system tries to identify a time (time slot in the electronic schedule) to which a task is not assigned to the resource, and presents the time for the new task on a display screen. However, if the new task needs to be urgently completed by an upcoming deadline, there may be no time vacant for the resource in the electronic schedule prior to the deadline of the new task. Alternatively, if the new task depends upon another (e.g., existing) task and needs to be performed within a (required) threshold period of time (e.g., within 3 hours of completion of the other task), there may be no time vacant (no free time slots) within the threshold period of time of the other task.

[0026] Thus, in order to complete the new task prior to the deadline or within the threshold period of time of the other task, one or more existing tasks may need to be moved in the electronic schedule. Unfortunately, a typical schedule may involve a large number of tasks that are linked with complex dependencies, which may often be difficult to sift through, assess and modify. Accordingly, users cannot themselves easily or accurately move existing tasks to different time slots to make a time available for a new task in accordance with the dependencies without

causing time conflicts. In one embodiment, the present scheduling system is configured to generate schedules for tasks to address this issue.

[0027] With reference to FIG. 1, one embodiment of a computerized system 100 associated with assigning a task to be performed using a resource at a candidate time is illustrated. The system 100 includes a controller 105, which may be configured to execute on a computer. The controller 105 may be initiated based upon a task request 110 being received from an entity computer 115, such as a computer of a vehicle-based delivery service that is requesting a task to be scheduled to be performed using a resource. For example, the task may be an appointment to deliver a first package to a loading bay, and pick up a second package from the loading bay. The controller 105 may receive the task request 110 over a network connection.

[0028] A scheduling data retriever 120 is configured to analyze the task request 110 and identify data within that identifies the task and associated parameters. The controller 105 utilizes the scheduling data retriever 120 to generate criteria 125 for existing task records to be considered for the task request 110. The criteria 125 may specify other tasks that depend upon the task, such as a second appointment to pick up the first package from the loading bay, and other tasks that the task depends upon, such as a third appointment to deliver the second package to the loading bay. In some examples, the criteria 125 may further specify information about the resource(s) to be used to perform the task, such as a first vehicle that is to deliver the first package and pick up the second package, and the loading bay.

[0029] The criteria 125 may specify a time frame within which the task needs to be scheduled. For example, the time frame may be determined based upon the task request 110, and may specify a starting point in time on or after which the task may be scheduled and an ending point in time on or prior to which the task may be scheduled. Alternatively, the time frame may specify the ending point in time

on or prior to which the task may be scheduled without specifying the starting point in time. For example, the criteria 125 may specify "within 3 days" as the time frame, in accordance with a need of a client of the vehicle-based delivery service to have the first package delivered (to a first recipient) and the second package delivered (to a second recipient) within 3 days.

[0030] The criteria 125 may further specify a threshold priority (level). The criteria 125 may further specify a medium priority as the threshold priority. For example, the task may have a medium priority, and the vehicle-based delivery service may not want to modify tasks with similar or higher priority when scheduling the task, and may instead only want to consider modifying tasks with a lower priority. In another example, a value of the client to the vehicle-based delivery service may be a medium value, and the vehicle-based delivery service may not want to modify tasks associated with customers that are of similar or higher value to the vehicle-based delivery service, and may instead only want to consider modifying tasks associated with customers that are of lower value to the vehicle-based delivery service.

[0031] In another example, the priority of the (new) task may be indicative of a level of (e.g., processor, memory, etc.) resources that are to be allocated to scheduling the task. For example, the higher the priority of the task, the greater the number of time slots and tasks of a scheduling data structure 130 that are to be considered when scheduling the task, and thus the more (e.g., processor, memory, etc.) resources are allocated to identify times (time slots in the scheduling data structure 130) and tasks when scheduling the task.

[0032] In some examples, a task that is to be performed using two resources is represented in the scheduling data structure 130 as two tasks. In the examples, a task that is to be performed using a vehicle and a loading bay is represented as a vehicle task and a loading bay task. In other examples, a task that is to be

performed using two resources is represented in the scheduling data structure 130 as a single task.

[0033] The criteria 125 is used to analyze the scheduling data structure 130 in order to identify task records 135 for a plurality of resources (vehicles, loading bays, etc.). The task records 135 may include the records for a first set of tasks in the scheduling data structure 130 that the resource (and/or the task) depends upon, and that are to be performed using other resources (different than the resource performing the task). For example, the third appointment for a second vehicle to deliver the second package to the loading bay may be identified in the scheduling data structure 130 for inclusion in the task records 135, as performance of the task by the first vehicle (in particular, picking up the second package from the loading bay) depends upon the second vehicle delivering the second package to the loading bay prior to arrival of the first vehicle at the loading bay. The task records 135 may also include the records for a second set of tasks in the scheduling data structure 130 that other resources depend upon, and that are to be performed using the resource (that is to perform the task). For example, the second appointment for a third vehicle to pick up the first package from the loading bay may be identified in the scheduling data structure 130 for inclusion in the task records 135, as picking up the first package by the third vehicle depends upon performance of the task by the first vehicle (in particular, delivering the first package to the loading bay) prior to arrival of the third vehicle at the loading bay.

[0034] The controller 105 utilizes the scheduling data analyzer 140 to analyze the task records 135 for the determination of dependencies between the first set of tasks and the second set of tasks. The task records 135 may each be considered to identify time(s) that the task can be performed. For example, the third appointment for the second vehicle to deliver the second package to the loading bay may be determined to be scheduled for 2 pm on May 3, while the

second appointment for the third vehicle to pick up the first package from the loading bay may be determined to be scheduled for 5 pm on May 3.

[0035] A determination may be made that if the task is scheduled for 12 pm on May 3, the first vehicle will be unable to pick up the second package, as required by the task, without delaying departure from the loading bay for a period of time exceeding a threshold or without returning to the loading bay a second time. A determination may also be made that if the task is scheduled for 6 pm on May 3, the third vehicle will be unable to pick up the first package, as required by the second appointment, without delaying departure from the loading bay for a period of time exceeding a threshold or without returning to the loading bay a second time. A determination may also be made that if the task is scheduled for 4 pm on May 3, the third vehicle will be able to pick up the first package, as required by the second appointment, without delay, and the first vehicle will be able to pick up the second package, as required by the task, without delay. Thus, 4 pm on May 3 may be calculated as being a candidate time for performing the task using the resource (the first vehicle). In one embodiment, the candidate time is calculated based upon a determination that the task will be performed within a first threshold period of time (3 hours) of performance of tasks that the task depends upon and/or within a second threshold period of time (2 hours) of performance of tasks that depend upon the task.

[0036] The controller 105 utilizes the scheduling data analyzer 140 to generate a candidate schedule for the plurality of resources identified in the task records 135. The candidate schedule includes the candidate time for the task to be performed using the resource. The candidate schedule may also include updated times for some tasks (deliveries, pick ups, etc.) to be performed using some resources and existing times for other tasks (deliveries, pick ups, etc.) to be performed using other resources. In some examples, the candidate schedule

includes calculated timings associated with tasks given certain assignments of the tasks and/or a sequence of the tasks.

[0037] The controller 105 utilizes the scheduling data analyzer 140 to evaluate the candidate schedule to calculate a candidate schedule performance score. For example, the candidate schedule performance score may be based upon a number of conflicts between tasks, a number of delays, and/or a number of repeat trips caused due to dependencies between tasks and/or resources predicted to result from implementation of the candidate schedule. In another example, the candidate schedule performance score may be based upon temporal flexibility (room for error) available between dependent tasks in the candidate schedule (e.g., 2 hours of temporal flexibility between the delivery of the second package to the loading bay by the second vehicle at 2pm and the task at 4 pm).

[0038] The controller 105 utilizes the scheduling data analyzer 140 to compare the candidate schedule performance score to an existing schedule performance score of an existing schedule (for the plurality of resources). The existing schedule may have already been implemented across the plurality of resources when the task request 110 is received by the controller 105. In response to determining that the candidate schedule performance score is greater than the existing schedule performance score, the scheduling data structure 130 is modified and regenerated, based upon the candidate schedule, to assign the task to be performed using the resource (the first vehicle) at the candidate time (4 pm on May 3).

[0039] If, instead, a determination is made that the candidate schedule performance score is not greater than the existing schedule performance score, the existing schedule is preserved, the candidate schedule is discarded, and a new candidate schedule (with a new candidate time for performing the task using the resource) is generated and evaluated. In some examples, the scheduling data structure 130 is only modified and regenerated based upon the candidate schedule

if the candidate schedule performance score is greater than the existing schedule performance score by a threshold margin.

[0040] In some examples, as part of the calculation of the candidate time and/or the generation of a candidate schedule, a proposed start time and a proposed completion time of each task of a plurality of a tasks is (repeatedly) recalculated based upon a completion time of a (previously calculated) predecessor task. The recalculation may be performed using an algorithm, which as part of (one or more iterations of) the calculation of the candidate time, the generation of a candidate schedule and/or the comparison of the candidate schedule performance score to the existing schedule performance score, cycles through one or more (e.g., all) tasks to recalculate a proposed completion time of each task based upon predecessor tasks. The algorithm continues cycling through the tasks until the times of the one or more tasks are no longer changed.

[0041] The controller 105 generates schedule instructions 145, including a portion of the regenerated scheduling data structure 130 corresponding to the candidate schedule, to provide to, and thus control, the schedule implementer 150. The schedule instructions 145 may thus provide for the schedule implementer 150 to instruct the first vehicle to deliver the first package to the loading bay, and pick up the second package from the loading bay at 4 pm on May 3 (the candidate time), instruct the second vehicle to deliver the second package to the loading bay at 2 pm on May 3, and instruct the third vehicle to pick up the first package from the loading bay at 5 pm on May 3.

[0042] It may be appreciated that tasks with higher priorities may be associated with a greater probability of not being movable to an alternative time slot than tasks with lower priorities. That is, in an example, even though it may be likely that a task with a priority higher than the threshold priority is movable to an alternative time slot, it is more likely that another task with a priority lower than the threshold priority is movable to an alternative time slot. Thus, it is more efficient to consider

the other task first. Using this heuristic approach, an appropriate time slot may be identified using less processing power, less memory usage, less database accessing, etc.

[0043] In one embodiment, the system 100 is a computing/data processing system including an application or collection of distributed applications for enterprise organizations. The applications and system 100 may be configured to operate with or be implemented as a cloud-based networking system, a software as a service (SaaS) architecture, or other type of networked computing solution. In one embodiment the system 100 is a centralized server-side application that provides at least the functions disclosed herein and that is accessed by many users via computing devices/terminals communicating with the system 100 (functioning as the server) over a computer network.

[0044] FIG. 2 illustrates one embodiment of a computer-implemented method 200 associated with assigning a first task to be performed using a first resource at a candidate time. In one embodiment, method 200 is performed by the controller 105 utilizing various computing resources of the computer 805 (shown in Fig. 8), such as the processor 810 for executing instructions, memory 815 and/or disks 830 for storing data structures within which control instructions are generated, and/or network hardware for transmitting data structures to remote computers over networks. The method 200 may be triggered based upon various triggers, such as receipt of the task request 110 from the entity computer 115, etc.

[0045] At 205, the task request 110 is received, and in response to receipt of the task request 110, a determination is made that the task request 110 is directed towards scheduling a task (digging a hole) to be performed using a resource (a first crew). In some examples, the task request 110 is received by a server (hosting the controller 105), from the entity computer 115.

[0046] Fig. 4 illustrates one embodiment of a graphical user interface 405 on the entity computer 115 that may be used to generate the task request 110. The

graphical user interface 405 is controlled to display a first graphical object 410, a second graphical object 415, a third graphical object 420, a fourth graphical object 425 and a fifth graphical object 430. The first graphical object 410 is configured to receive user input that selects general information, such as a location, of the first task. The second graphical object 415 is configured to receive user input that selects dependency information of the first task, such as an indication of the first task being dependent upon another task. The third graphical object 420 is configured to receive user input that selects resource information about the first resource (the first crew) that is to perform the first task (digging the hole). The fourth graphical object 425 is configured to receive user input that selects a time frame (1 month, 1 season, 1 year, etc.) within which the first task is to be performed.

[0047] Returning to Fig. 2, at 210, the scheduling data structure 130 is accessed from a database via a network communication. One embodiment of the scheduling data structure 130 is illustrated in Fig. 3 and includes indications of a plurality of tasks 305 and a plurality of resources 310, assigned time slots 315, tasks depended upon 320 and dependent tasks 325 associated with one or more of the plurality of tasks 305. Task records 135 in the scheduling data structure 130 that correspond to the criteria 125 may be accessed (downloaded), while other task records that do not correspond to the criteria 125 may not be accessed.

[0048] The task records 135 that are accessed include a first set of tasks, upon which the first resource (the first crew) depends, to be performed using one or more other resources (other crews). For example, Fig. 3 indicates that the first task (digging the hole) is to be performed using the first resource (the first crew), and that the first task depends upon a second task (marking a location for the hole to be dug) that is to be performed using a second resource (a second crew). Accordingly, the second task is among the first set of tasks of the task records 135 that are accessed.

[0049] The task records 135 that are accessed also include a second set of tasks, upon which other resources depend, to be performed using the first resource. For example, Fig. 3 indicates that the first task (digging the hole) is depended upon by a third task (laying wire into the hole) that is to be performed using a third resource (a third crew). Accordingly, the third task is among the second set of tasks of the task records 135 that are accessed.

[0050] Returning to Fig. 2, at 215, the task records 135 are analyzed to determine dependencies between the first set of tasks to be performed using the one or more other resources and the second set of tasks to be performed using the first resource. For example, the dependencies between the first task (digging the hole), the second task (marking the location for the hole to be dug), the third task (laying wire into the hole), and other associated tasks, such as the dependence of the second task on performance of a fourth task (delivering of equipment to be used in the marking to the location) to be performed by a fourth resource (a fourth crew), may be determined.

[0051] At 220, a candidate time for performing the first task using the first resource is calculated based upon the dependencies. For example, a candidate time of 4 pm on November 1, 2016 is determined for the first task (digging the hole) based upon the times and dependencies of the second task (marking the location for the hole to be dug), the third task (laying wire into the hole), and the fourth task (delivering of equipment to be used in the marking to the location). For example, the candidate time for performing the first task using the first resource may be determined to be a time after one or more tasks that the first task depends upon are to be performed, and a time before one or more tasks that depend upon the first task are to be performed. In one embodiment, the candidate time is calculated based upon a determination that the first task will be performed within a first threshold period of time of performance of tasks that the first task depends upon

and/or within a second threshold period of time of performance of tasks that depend upon the first task.

[0052] At 225, a candidate schedule for the plurality of resources, including the first resource, is generated using the candidate time for the first task for the first resource. For example, the candidate schedule may assign the candidate time of 4 pm on November 1, 2016 to the first task (digging the hole), may maintain the existing times for the second task, the third task, and the fourth task, and may assign updated times to one or more other tasks scheduled to be performed by the plurality of resources.

[0053] At 230, the candidate schedule is evaluated to calculate a candidate schedule performance score. In some examples, the candidate schedule performance score is based upon a number of conflicts identified (or estimated) in dependencies between the first set of tasks and the second set of tasks in the candidate schedule. Conflicts are identified where a task that depends upon another task is scheduled to be performed before the other task. For example, a conflict is determined for a fifth task that is to be performed using the first resource if the fifth task has a time before a sixth task to be performed using a sixth resource, wherein the dependencies comprise a dependency of the fifth task on the sixth task. In another example, a conflict is determined for a seventh task to be performed using the sixth resource if the seventh task has a time before an eighth task to be performed using the first resource, wherein the dependencies comprise a dependency of the seventh task on the eighth task.

[0054] Returning to Fig. 4, the fifth graphical object 430 is configured to display information about the candidate schedule generated based upon the user input received via graphical objects 410, 415, 420 and/or 425. The fifth graphical object 430 includes a sixth graphical object 435 configured to display information about the candidate schedule, a seventh graphical object 440 configured to display the candidate schedule performance score calculated for the candidate schedule, and

an eighth graphical object 445 configured to display a comparison of the candidate schedule performance score calculated for the candidate schedule with an existing schedule performance score calculated for an existing schedule.

[0055] Returning to Fig. 2, at 235, in response to determining that the candidate schedule performance score is greater than the existing schedule performance score of the existing schedule, the scheduling data structure 130 is modified and regenerated, based upon the candidate schedule, to assign the first task to be performed using the first resource at the candidate time. For example, the performance of the first task using the first resource is assigned to a candidate time slot, of the candidate time, in the scheduling data structure 130. The assignment of the first task to the candidate time slot in the scheduling data structure 130 causes the first task to be performed by the first resource at the candidate time. In some examples, the scheduling data structure 130 (or the portion of the scheduling data structure 130 that is modified) is included in the schedule instructions 145, which are provided to the schedule implementer 150.

[0056] In some examples, one or more of the acts of Fig. 2 are iteratively repeated for various resources or tasks to test modifications to the schedule, incorporate the modifications that are determined to improve the schedule (by reducing conflict), and discard the modifications that are determined to not improve the schedule.

[0057] For example, a second candidate time for performing the second task using the second resource (different than an existing second time for the second task) is calculated based upon one or more dependencies identified between the tasks to be performed using the plurality of resources. For example, the second candidate time for performing the second task using the second resource may be determined to be a time after one or more tasks that the second task depends upon are to be performed, and a time before one or more tasks that depend upon the second task are to be performed.

[0058] A second candidate schedule for the plurality of resources, including the second resource, is generated using the second candidate time for the second task for the second resource. For example, the second candidate schedule may maintain the candidate time for the first task and the existing times for the third task and the fourth task, and may assign updated times to one or more other tasks scheduled to be performed by the plurality of resources.

[0059] The second candidate schedule is evaluated to calculate a second candidate schedule performance score. In some examples, the second candidate schedule performance score is based upon a number of conflicts identified (or estimated) in dependencies between the tasks to be performed using the plurality of resources in the second candidate schedule.

[0060] In response to determining that the second candidate schedule performance score is not greater than a second existing schedule performance score of a second existing schedule, a third candidate time for performing the second task using the second resource is calculated based upon the one or more dependencies. For example, the second candidate time and the second candidate schedule are determined to not improve the schedule, and are thus discarded, and the third candidate time is calculated for performing the second task using the second resource.

[0061] In response to determining that the second candidate schedule performance score is greater than the second existing schedule performance score of the second existing schedule, the scheduling data structure is modified and regenerated, based upon the second candidate schedule, to assign the second task to be performed using the second resource at the second candidate time. For example, the second candidate time and the second candidate schedule are determined to improve the schedule, and are thus implemented. For example, the performance of the second task using the second resource is assigned to a

second candidate time slot, of the second candidate time, in the scheduling data structure 130.

[0062] The schedule implementer 150 uses the scheduling data structure 130 to cause the first task to be performed by the first resource at the candidate time, and cause one or more other tasks to be scheduled for performance at other times. In some examples, the schedule implementer 150 may provide physical/print instructions to an operator of the resources that are to be used to perform the tasks, while in other examples, the schedule implementer 150 may provide digital instructions to one or more machines configured to control the resources, such as computers of the crews or their vehicles, in accordance with the schedule instructions 145.

[0063] One embodiment of the schedule instructions 145 is shown in Fig. 5, as a data structure received by the promotional display implementer 150. The schedule instructions 145 include a first instruction 510 to assign the first task to be performed using the first resource at the candidate time, a second instruction 515 to unassign the second task to be performed using the second resource from a second time, and a third instruction 520 to assign the second task to be performed using the second resource at a second candidate time (calculated for the second task based upon the dependencies).

[0064] It may be appreciated that in some examples, the schedule implementer 150 monitors the status of one or more resources, and the implementation of the candidate schedule, as illustrated in Fig. 6. The schedule implementer 150 includes a monitor interface 605, which displays a first indicator 610 displaying a status of the first task to be performed using the first resource, and a second indicator 615 displaying a status of the second task to be performed using the second resource. For example, the first indicator 610 may indicate that the first resource (the first crew) has incorporated the first task into its schedule at the candidate time in accordance with the schedule instructions 145, while the second

indicator may indicate that the second resource (the second crew) has not yet incorporated the second task into its schedule at the second candidate time, and is thus not in accordance with the schedule instructions 145. In some examples, the monitor interface 605 is generated based upon user feedback indicative of impressions of users associated with the resources, while in other examples, the monitor interface 605 is generated based upon automatic analysis of the respective schedules of the resources.

[0065] In one embodiment, an algorithm used to perform at least some of the techniques disclosed herein collects one or more resources involved in a schedule, clears an event table corresponding to the schedule, processes the one or more resources collected in a first loop, calculates times for first tasks assigned to a first resource of the one or more resources, adds information about the first tasks to the event table, calculates times for second tasks assigned to a second resource of the one or more resources while taking into account the first tasks added to the event table if the second resource depends upon at least one of the first tasks, adds information about the second tasks to the event table. The algorithm iteratively repeats (e.g., hundreds of times per second) this process to gradually generate a schedule with correct (non-conflicting) dependencies and tasks. It may be appreciated that the event table may be a hash table containing estimated times for one or more tasks, and events needed to satisfy dependencies between a plurality of tasks.

[0066] Fig. 7 is an illustration of a scenario 700 involving an example non-transitory computer-readable medium 705. In one embodiment, one or more of the components described herein are configured as program modules, such as the controller 105, stored in the non-transitory computer-readable medium 705. The program modules are configured with stored instructions, such as processor-executable instructions 710, that when executed by at least a processor, such as processor 715, cause the computing device to perform the corresponding

function(s) as described herein. For example, functionality of the controller 105, stored in the non-transitory computer-readable medium 705, may be executed by the processor 715 as the processor-executable instructions 710 to perform an embodiment 740 of the method 200 of Fig. 2.

[0067] The non-transitory machine readable medium 705 includes the processor-executable instructions 710 that when executed by a processor 715 cause performance of at least some of the provisions herein. The non-transitory machine readable medium 705 includes a memory semiconductor (e.g., a semiconductor utilizing static random access memory (SRAM), dynamic random access memory (DRAM), and/or synchronous dynamic random access memory (SDRAM) technologies), a platter of a hard disk drive, a flash memory device, or a magnetic or optical disc (such as a compact disk (CD), a digital versatile disk (DVD), or floppy disk). The example non-transitory machine readable medium 705 stores computer-readable data 720 that, when subjected to reading 725 by a reader 730 of a device 735 (e.g., a read head of a hard disk drive, or a read operation invoked on a solid-state storage device), express the processor-executable instructions 710. In some embodiments, the processor-executable instructions 710, when executed cause performance of operations, such as at least some of the example method 200 of Fig. 2, for example. In some embodiments, the processor-executable instructions 710 are configured to cause implementation of a system, such as at least some of the example system 100 of Fig. 1, for example.

[0068] FIG. 8 illustrates a scenario 800 of an example computing device that is configured and/or programmed with one or more of the example systems and methods described herein, and/or equivalents. The example computing device may be a computer 805 that includes a processor 810, a memory 815, and input/output ports 820 operably connected by a bus 825. In one example, the computer 805 may include logic of the controller 105 configured to facilitate the

system 100 and/or the method 200 shown in Figs. 1 and 2. In different examples, the logic of the controller 105 may be implemented in hardware, a non-transitory computer-readable medium 705 with stored instructions, firmware, and/or combinations thereof. While the logic of the controller 105 is illustrated as a hardware component attached to the bus 825, it is to be appreciated that in other embodiments, the logic of the controller 105 could be implemented in the processor 810, stored in memory 815, or stored in disk 830.

[0069] In one embodiment, logic of the controller 105 or the computer 805 is a means (e.g., structure: hardware, non-transitory computer-readable medium, firmware) for performing the actions described. In some embodiments, the computing device may be a server operating in a cloud computing system, a server configured in a Software as a Service (SaaS) architecture, a smart phone, laptop, tablet computing device, and so on.

[0070] The means may be implemented, for example, as an application specific integrated circuit (ASIC) programmed to implement rule based source sequencing for allocation. The means may also be implemented as stored computer executable instructions that are presented to computer 805 as data 845 that are temporarily stored in memory 815 and then executed by processor 810.

[0071] The logic of the controller 105 may also provide means (e.g., hardware, non-transitory computer-readable medium 705 that stores executable instructions, firmware) for performing rule based source sequencing for allocation.

[0072] Generally describing an example configuration of the computer 805, the processor 810 may be a variety of various processors including dual microprocessor and other multi-processor architectures. The memory 815 may include volatile memory and/or non-volatile memory. Non-volatile memory may include, for example, read-only memory (ROM), programmable read-only memory (PROM), and so on. Volatile memory may include, for example, random access

memory (RAM), static random-access memory (SRAM), dynamic random access memory (DRAM), and so on.

[0073] The disks 830 may be operably connected to the computer 805 via, for example, an input/output (I/O) interface (e.g., card, device) 835 and an input/output port 820. The disks 830 may be, for example, a magnetic disk drive, a solid state disk drive, a floppy disk drive, a tape drive, a Zip drive, a flash memory card, a memory stick, and so on. Furthermore, the disks 830 may be a CD-ROM drive, a CD-R drive, a CD-RW drive, a DVD ROM, and so on. The memory 815 can store a process 840 and/or a data 845, for example. The disk 830 and/or the memory 815 can store an operating system that controls and allocates resources of the computer 805.

[0074] The computer 805 may interact with input/output (I/O) devices via the I/O interfaces 835 and the input/output ports 820. Input/output devices may be, for example, a keyboard, a microphone, a pointing and selection device, cameras, video cards, displays, the disks 830, the network devices 850, and so on. The input/output ports 820 may include, for example, serial ports, parallel ports, and USB ports. I/O controllers 855 may connect the I/O interfaces 835 to the bus 825.

[0075] The computer 805 can operate in a network environment and thus may be connected to the network devices 850 via the I/O interfaces 835, and/or the I/O ports 820. Through the network devices 850, the computer 805 may interact with a network. Through the network, the computer 805 may be logically connected to remote computers. Networks with which the computer 805 may interact include, but are not limited to, a local area network (LAN), a new area network (WAN), and other networks.

[0076] In another embodiment, the described methods and/or their equivalents may be implemented with computer executable instructions. Thus, in one embodiment, a non-transitory computer readable/storage medium is configured with stored computer executable instructions of an algorithm/executable

application that when executed by a machine(s) cause the machine(s) (and/or associated components) to perform the method. Example machines include but are not limited to a processor, a computer, a server operating in a cloud computing system, a server configured in a Software as a Service (SaaS) architecture, a smart phone, and so on). In one embodiment, a computing device is implemented with one or more executable algorithms that are configured to perform any of the disclosed methods.

[0077] In one or more embodiments, the disclosed methods or their equivalents are performed by either: computer hardware configured to perform the method; or computer instructions embodied in a module stored in a non-transitory computer-readable medium where the instructions are configured as an executable algorithm configured to perform the method when executed by at least a processor of a computing device.

[0078] While for purposes of simplicity of explanation, the illustrated methodologies in the figures are shown and described as a series of blocks of an algorithm, it is to be appreciated that the methodologies are not limited by the order of the blocks. Some blocks can occur in different orders and/or concurrently with other blocks from that shown and described. Moreover, less than all the illustrated blocks may be used to implement an example methodology. Blocks may be combined or separated into multiple actions/components. Furthermore, additional and/or alternative methodologies can employ additional actions that are not illustrated in blocks.

[0079] The following includes definitions of selected terms employed herein. The definitions include various examples and/or forms of components that fall within the scope of a term and that may be used for implementation. The examples are not intended to be limiting. Both singular and plural forms of terms may be within the definitions.

[0080] References to "one embodiment", "an embodiment", "one example", "an example", and so on, indicate that the embodiment(s) or example(s) so described may include a particular feature, structure, characteristic, property, element, or limitation, but that not every embodiment or example necessarily includes that particular feature, structure, characteristic, property, element or limitation. Furthermore, repeated use of the phrase "in one embodiment" does not necessarily refer to the same embodiment, though it may.

[0081] A "data structure", as used herein, is an organization of data in a computing system that is stored in a memory, a storage device, or other computerized system. A data structure may be any one of, for example, a data field, a data file, a data array, a data record, a database, a data table, a graph, a tree, a linked list, and so on. A data structure may be formed from and contain many other data structures (e.g., a database includes many data records). Other examples of data structures are possible as well, in accordance with other embodiments.

[0082] "Computer-readable medium" or "computer storage medium", as used herein, refers to a non-transitory medium that stores instructions and/or data configured to perform one or more of the disclosed functions when executed. Data may function as instructions in some embodiments. A computer-readable medium may take forms, including, but not limited to, non-volatile media, and volatile media. Non-volatile media may include, for example, optical disks, magnetic disks, and so on. Volatile media may include, for example, semiconductor memories, dynamic memory, and so on. Common forms of a computer-readable medium may include, but are not limited to, a floppy disk, a flexible disk, a hard disk, a magnetic tape, other magnetic medium, an application specific integrated circuit (ASIC), a programmable logic device, a compact disk (CD), other optical medium, a random access memory (RAM), a read only memory (ROM), a memory chip or card, a memory stick, solid state storage device (SSD), flash drive, and other media from

which a computer, a processor or other electronic device can function with. Each type of media, if selected for implementation in one embodiment, may include stored instructions of an algorithm configured to perform one or more of the disclosed and/or claimed functions.

[0083] "Logic", as used herein, represents a component that is implemented with computer or electrical hardware, a non-transitory medium with stored instructions of an executable application or program module, and/or combinations of these to perform any of the functions or actions as disclosed herein, and/or to cause a function or action from another logic, method, and/or system to be performed as disclosed herein. Equivalent logic may include firmware, a microprocessor programmed with an algorithm, a discrete logic (e.g., ASIC), at least one circuit, an analog circuit, a digital circuit, a programmed logic device, a memory device containing instructions of an algorithm, and so on, any of which may be configured to perform one or more of the disclosed functions. In one embodiment, logic may include one or more gates, combinations of gates, or other circuit components configured to perform one or more of the disclosed functions. Where multiple logics are described, it may be possible to incorporate the multiple logics into one logic. Similarly, where a single logic is described, it may be possible to distribute that single logic between multiple logics. In one embodiment, one or more of these logics are corresponding structure associated with performing the disclosed and/or claimed functions. Choice of which type of logic to implement may be based on desired system conditions or specifications. For example, if greater speed is a consideration, then hardware would be selected to implement functions. If a lower cost is a consideration, then stored instructions/executable application would be selected to implement the functions.

[0084] An "operable connection", or a connection by which entities are "operably connected", is one in which signals, physical communications, and/or logical communications may be sent and/or received. An operable connection may

include a physical interface, an electrical interface, and/or a data interface. An operable connection may include differing combinations of interfaces and/or connections sufficient to allow operable control. For example, two entities can be operably connected to communicate signals to each other directly or through one or more intermediate entities (e.g., processor, operating system, logic, non-transitory computer-readable medium). Logical and/or physical communication channels can be used to create an operable connection.

[0085] "User", as used herein, includes but is not limited to one or more persons, computers or other devices, or combinations of these.

[0086] While the disclosed embodiments have been illustrated and described in considerable detail, it is not the intention to restrict or in any way limit the scope of the appended claims to such detail. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the various aspects of the subject matter. Therefore, the disclosure is not limited to the specific details or the illustrative examples shown and described. Thus, this disclosure is intended to embrace alterations, modifications, and variations that fall within the scope of the appended claims.

[0087] To the extent that the term "includes" or "including" is employed in the detailed description or the claims, it is intended to be inclusive in a manner similar to the term "comprising" as that term is interpreted when employed as a transitional word in a claim.

[0088] To the extent that the term "or" is used in the detailed description or claims (e.g., A or B) it is intended to mean "A or B or both". When the applicants intend to indicate "only A or B but not both" then the phrase "only A or B but not both" will be used. Thus, use of the term "or" herein is the inclusive, and not the exclusive use.

CLAIMS

What is claimed is:

1. A computing system, comprising:
a processor connected to memory; and
a scheduling module stored on a non-transitory computer readable medium and configured with instructions that when executed by the processor cause the processor to:
in response to receiving a request to modify a scheduling data structure that includes a plurality of tasks, determine that the request is directed towards scheduling a first task to be performed using a first resource from a plurality of resources;
access the scheduling data structure from a database via a network communication, wherein the scheduling data structure includes data records for:
(i) a first set of tasks, upon which the first resource depends, to be performed using one or more other resources, and
(ii) a second set of tasks, upon which other resources depend, to be performed using the first resource;
analyze the data records to determine dependencies between the first set of tasks to be performed using the one or more other resources and the second set of tasks to be performed using the first resource;
calculate a candidate time for performing the first task using the first resource based upon the dependencies;

generate a candidate schedule for the plurality of resources, including the first resource, using the candidate time for the first task for the first resource;

evaluate the candidate schedule to calculate a candidate schedule performance score; and

in response to determining that the candidate schedule performance score is greater than an existing schedule performance score of an existing schedule, modify and regenerate the scheduling data structure, based upon the candidate schedule, to assign the first task to be performed using the first resource at the candidate time.

2. The computing system of claim 1,

wherein the instructions when executed by the processor cause the processor to iteratively:

calculate a second candidate time for performing a second task using a second resource, from the plurality of resources, based upon one or more dependencies;

generate a second candidate schedule for the plurality of resources, including the second resource, using the second candidate time for the second task for the second resource;

evaluate the second candidate schedule to calculate a second candidate schedule performance score;

in response to determining that the second candidate schedule performance score is not greater than a second existing schedule performance score of a second existing schedule, calculate a third candidate time for performing the second task using the second resource, from the plurality of resources, based upon the one or more dependencies; and

in response to determining that the second candidate schedule performance score is greater than the second existing schedule performance score of the second existing schedule, modify and regenerate the scheduling data structure, based upon the second candidate schedule, to assign the second task to be performed using the second resource at the second candidate time.

3. The computing system of claims 1 or 2,

wherein the instructions to evaluate the candidate schedule to calculate the candidate schedule performance score further include instructions that when executed by the processor cause the processor to:

analyze the candidate schedule to determine a number of conflicts in dependencies between the first set of tasks and the second set of tasks in the candidate schedule; and

generate the candidate schedule performance score based upon the number of conflicts in dependencies.

4. The computing system of claim 3, wherein the conflicts in dependencies comprise at least one of:

(i) a conflict corresponding to a second task to be performed using the first resource having a time before a third task to be performed using a second resource, wherein the dependencies comprise a dependency of the second task to be performed using the first resource on the third task to be performed using the second resource; or

(ii) a conflict corresponding to a fourth task to be performed using the second resource having a time before a fifth task to be performed

using the first resource, wherein the dependencies comprise a dependency of the fourth task to be performed using the second resource on the fifth task to be performed using the first resource.

5. The computing system of claims 1, 2 or 3,

wherein the dependencies used to calculate the candidate time for performing the first task using the first resource comprise at least one of:

(i) a dependency of the first task to be performed using the first resource on a second task to be performed using a second resource; or

(ii) a dependency of a third task to be performed using the second resource on the first task to be performed using the first resource; and

wherein the instructions to calculate the candidate time for performing the first task using the first resource based upon the dependencies further include instructions that when executed by the processor cause the processor to at least one of:

calculate the candidate time to be after a time of the second task based upon the dependency of the first task to be performed using the first resource on the second task to be performed using the second resource; or

calculate the candidate time to be before a time of the third task based upon the dependency of the third task to be performed using the second resource on the first task to be performed using the first resource.

6. The computing system of claims 1, 2, 3, or 5,

wherein the dependencies used to calculate the candidate time for performing the first task using the first resource comprise at least one of:

(i) a dependency of a second task for the first resource on a third task for a second resource; or

(ii) a dependency of a fourth task for the second resource on a fifth task for the first resource;

wherein the instructions when executed by the processor cause the processor to at least one of:

calculate a second candidate time for the second task for the first resource to be after a time of the third task based upon the dependency of the second task for the first resource on the third task for the second resource; or

calculate a third candidate time for the fifth task for the first resource to be before a time of the fourth task based upon the dependency of the fourth task for the second resource on the fifth task for the first resource; and

wherein the instructions to generate the candidate schedule for the plurality of resources further include instructions that when executed by the processor cause the processor to at least one of:

use the second candidate time for the second task for the first resource in the candidate schedule; or

use the third candidate time for the fifth task for the first resource in the candidate schedule.

7. The computing system of any one of claims 1-6,

wherein the first resource is a first vehicle, the first set of tasks are to be performed using one or more other vehicles, and the second set of tasks are to be performed by the first vehicle;

wherein the instructions to determine that the candidate schedule performance score is greater than the existing schedule performance score of the existing schedule further include instructions that when executed by the processor cause the processor to:

determine that the candidate schedule resolves a conflict in the existing schedule caused by a dependency between the first vehicle and at least one of the one or more other vehicles.

8. A computer-implemented method performed by a computing device comprising a processor, the computer-implemented method comprising:

receiving, by at least the processor, a request to modify a scheduling data structure that includes a plurality of tasks;

accessing, by at least the processor, the scheduling data structure from a database via a network communication, wherein the scheduling data structure includes data records for:

(i) a first set of tasks, upon which a first resource depends, to be performed using one or more other resources, and

(ii) a second set of tasks, upon which other resources depend, to be performed using the first resource;

analyzing, by at least the processor, the data records to determine dependencies between the first set of tasks to be performed using the one or more other resources and the second set of tasks to be performed using the first resource;

calculating, by at least the processor, a candidate time for performing a first task using the first resource based upon the dependencies;

generating, by at least the processor, a candidate schedule for the plurality of resources, including the first resource, using the candidate time for the first task for the first resource;

evaluating, by at least the processor, the candidate schedule to calculate a candidate schedule performance score; and

in response to determining that the candidate schedule performance score is greater than an existing schedule performance score of an existing schedule, modifying and regenerating, by at least the processor, the scheduling data structure, based upon the candidate schedule, to assign the first task to be performed using the first resource at the candidate time.

9. The computer-implemented method of claim 8, further comprising iteratively:

calculating a second candidate time for performing a second task using a second resource, from the plurality of resources, based upon one or more dependencies;

generating a second candidate schedule for the plurality of resources, including the second resource, using the second candidate time for the second task for the second resource;

evaluating the second candidate schedule to calculate a second candidate schedule performance score;

in response to determining that the second candidate schedule performance score is not greater than a second existing schedule performance score of a second existing schedule, calculating a third candidate time for performing the second task using the

second resource, from the plurality of resources, based upon the one or more dependencies; and

in response to determining that the second candidate schedule performance score is greater than the second existing schedule performance score of the second existing schedule, modifying and regenerating the scheduling data structure, based upon the second candidate schedule, to assign the second task to be performed using the second resource at the second candidate time.

10. The computer-implemented method of claims 8 or 9,

wherein the evaluating the candidate schedule to calculate the candidate schedule performance score further comprises:

analyzing the candidate schedule to determine a number of conflicts in dependencies between the first set of tasks and the second set of tasks in the candidate schedule; and

generating the candidate schedule performance score based upon the number of conflicts in dependencies.

11. The computer-implemented method of any of claims 8-10, wherein the conflicts in dependencies comprise at least one of:

(i) a conflict corresponding to a second task to be performed using the first resource having a time before a third task to be performed using a second resource, wherein the dependencies comprise a dependency of the second task to be performed using the first resource on the third task to be performed using the second resource; or

(ii) a conflict corresponding to a fourth task to be performed using the second resource having a time before a fifth task to be performed using the first resource, wherein the dependencies comprise a

dependency of the fourth task to be performed using the second resource on the fifth task to be performed using the first resource.

12. The computer-implemented method of claims 8, 9, or 10,
wherein the dependencies used to calculate the candidate time for performing the first task using the first resource comprise at least one of:
(i) a dependency of the first task to be performed using the first resource on a second task to be performed using a second resource; or
(ii) a dependency of a third task to be performed using the second resource on the first task to be performed using the first resource; and
wherein the calculating the candidate time for performing the first task using the first resource based upon the dependencies further comprises at least one of:
calculating the candidate time to be after a time of the second task based upon the dependency of the first task to be performed using the first resource on the second task to be performed using the second resource; or
calculating the candidate time to be before a time of the third task based upon the dependency of the third task to be performed using the second resource on the first task to be performed using the first resource.

13. The computer-implemented method of claims 8, 9, 10 or 12,
wherein the dependencies used to calculate the candidate time for performing the first task using the first resource comprise at least one of:

(i) a dependency of a second task for the first resource on a third task for a second resource; or

(ii) a dependency of a fourth task for the second resource on a fifth task for the first resource.

14. The computer-implemented method of claim 13,

wherein the computer-implemented method further comprises at least one of:

calculating a second candidate time for the second task for the first resource to be after a time of the third task based upon the dependency of the second task for the first resource on the third task for the second resource; or

calculating a third candidate time for the fifth task for the first resource to be before a time of the fourth task based upon the dependency of the fourth task for the second resource on the fifth task for the first resource; and

15. The computer-implemented method of claim 14,

wherein the generating the candidate schedule for the plurality of resources further comprises at least one of:

using the second candidate time for the second task for the first resource in the candidate schedule; or

using the third candidate time for the fifth task for the first resource in the candidate schedule.

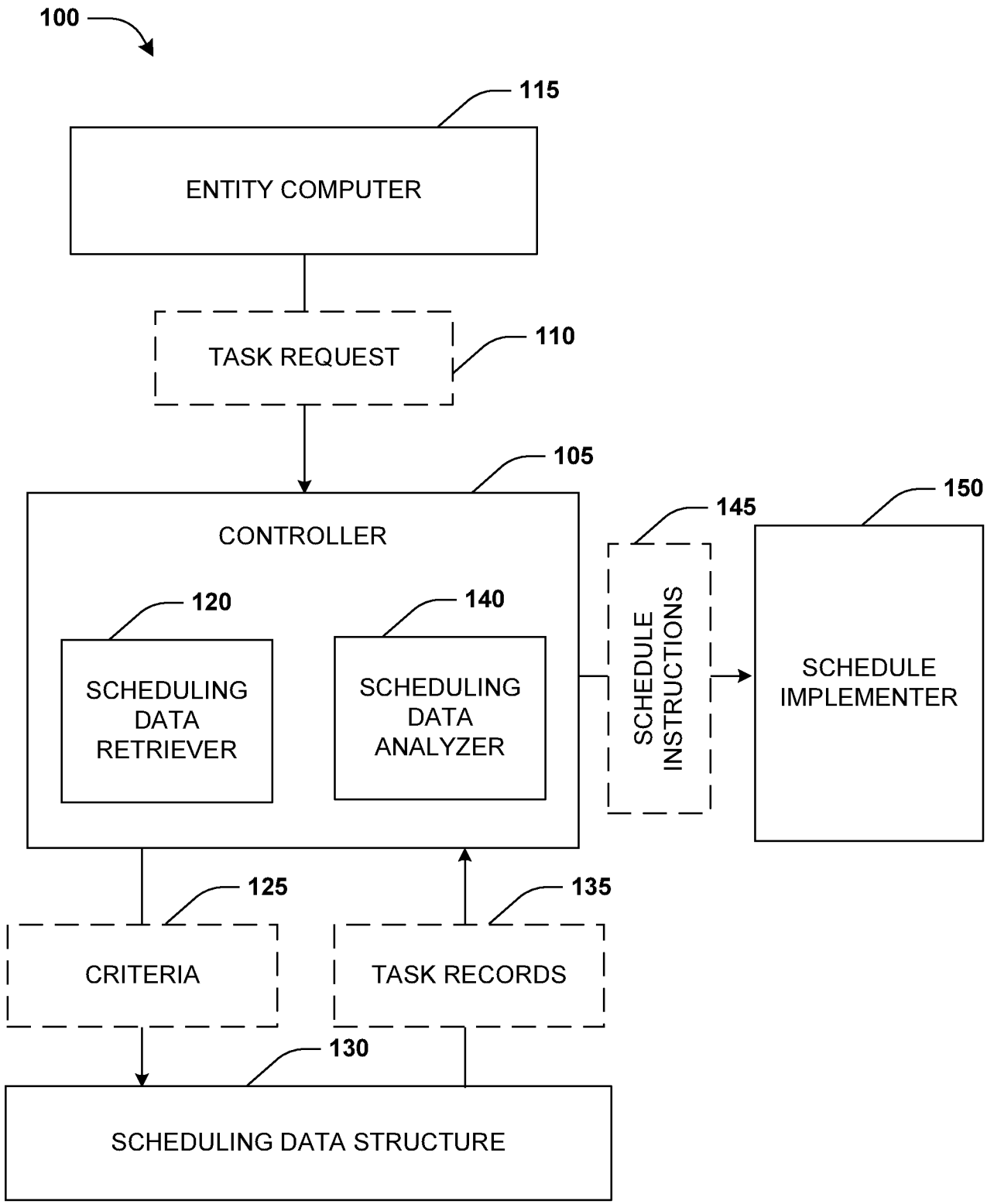
100
115
ENTITY COMPUTER
TASK REQUEST
110
105
CONTROLLER
120
SCHEDULING DATA RETRIEVER
140
SCHEDULING DATA ANALYZER
145
SCHEDULE INSTRUCTIONS
150
SCHEDULE IMPLEMENTER
125
CRITERIA
135
TASK RECORDS
130
SCHEDULING DATA STRUCTURE

FIG. 1

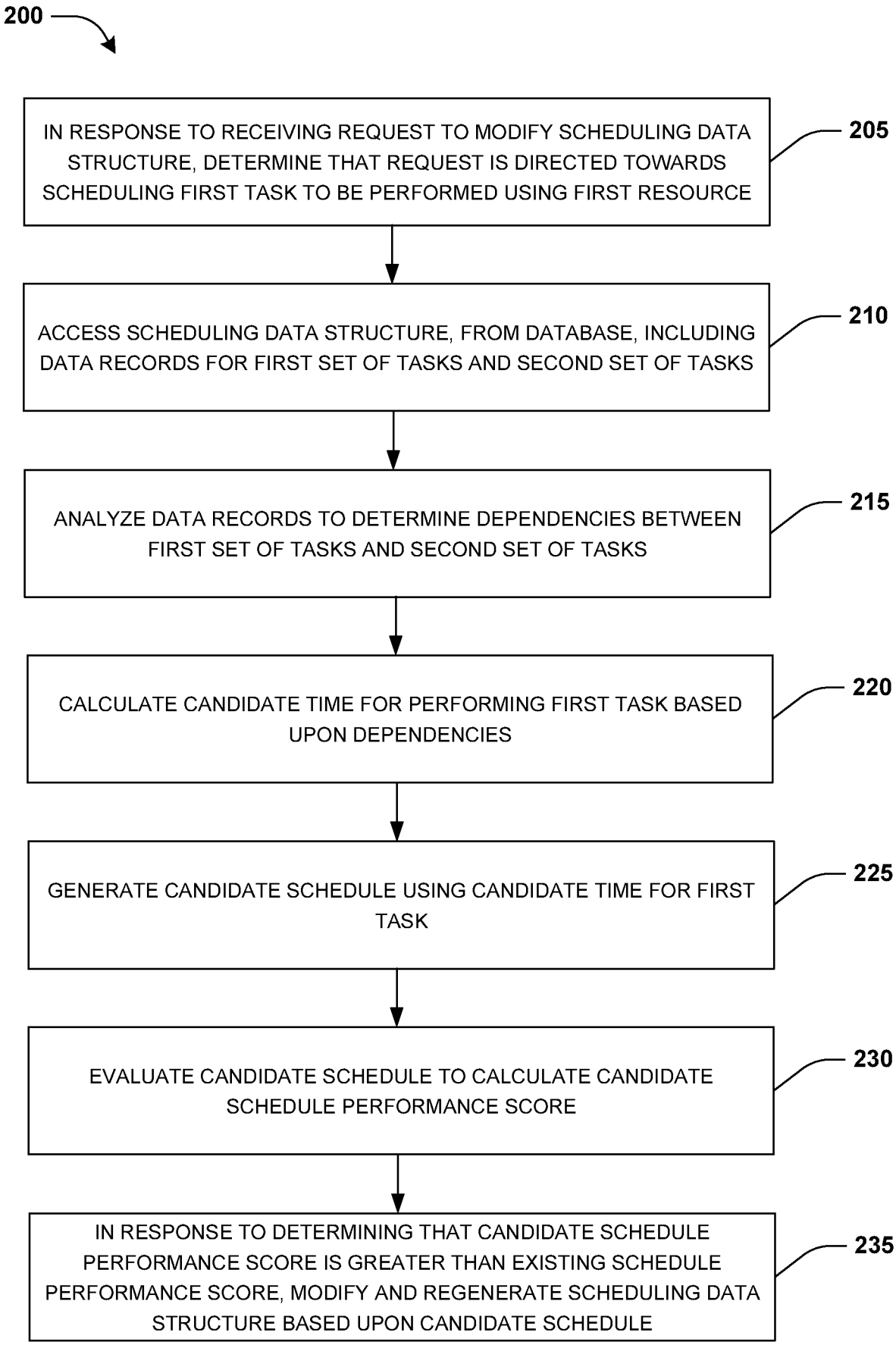
200
IN RESPONSE TO RECEIVING REQUEST TO MODIFY SCHEDULING DATA STRUCTURE, DETERMINE THAT REQUEST IS DIRECTED TOWARDS SCHEDULING FIRST TASK TO BE PERFORMED USING FIRST RESOURCE
205
ACCESS SCHEDULING DATA STRUCTURE, FROM DATABASE, INCLUDING DATA RECORDS FOR FIRST SET OF TASKS AND SECOND SET OF TASKS
210
ANALYZE DATA RECORDS TO DETERMINE DEPENDENCIES BETWEEN FIRST SET OF TASKS AND SECOND SET OF TASKS
215
CALCULATE CANDIDATE TIME FOR PERFORMING FIRST TASK BASED UPON DEPENDENCIES
220
GENERATE CANDIDATE SCHEDULE USING CANDIDATE TIME FOR FIRST TASK
225
EVALUATE CANDIDATE SCHEDULE TO CALCULATE CANDIDATE SCHEDULE PERFORMANCE SCORE
230
IN RESPONSE TO DETERMINING THAT CANDIDATE SCHEDULE PERFORMANCE SCORE IS GREATER THAN EXISTING SCHEDULE PERFORMANCE SCORE, MODIFY AND REGENERATE SCHEDULING DATA STRUCTURE BASED UPON CANDIDATE SCHEDULE
235

FIG. 2

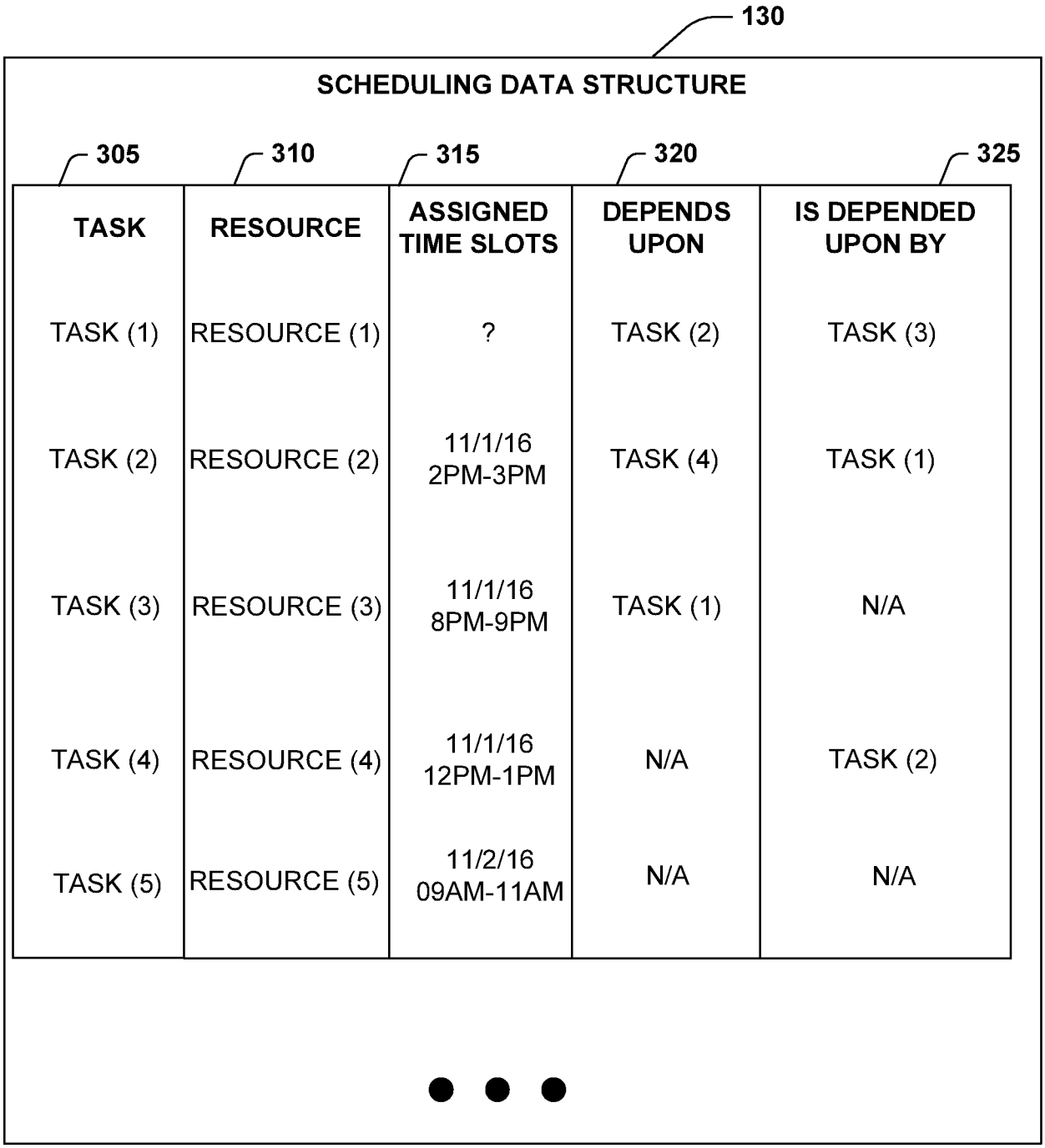
130
SCHEDULING DATA STRUCTURE
305
310
315
320
325
TASK
RESOURCE
ASSIGNED TIME SLOTS
DEPENDS UPON
IS DEPENDED UPON BY
TASK (1)
RESOURCE (1)
?
TASK (2)
TASK (3)
TASK (2)
RESOURCE (2)
11/1/16 2PM-3PM
TASK (4)
TASK (1)
TASK (3)
RESOURCE (3)
11/1/16 8PM-9PM
TASK (1)
N/A
TASK (4)
RESOURCE (4)
11/1/16 12PM-1PM
N/A
TASK (2)
TASK (5)
RESOURCE (5)
11/2/16 09AM-11AM
N/A
N/A

FIG. 3

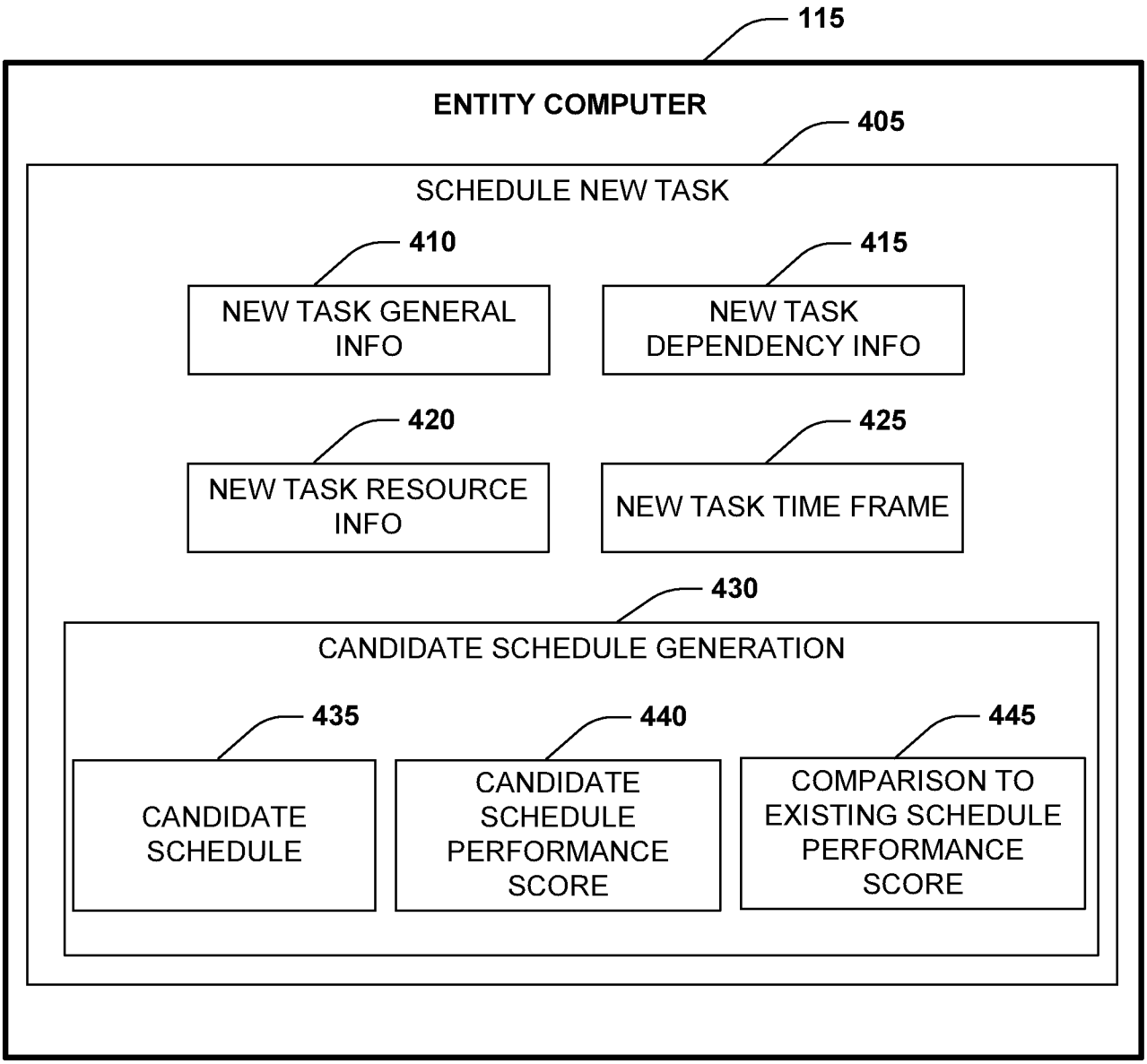
115
ENTITY COMPUTER
405
SCHEDULE NEW TASK
410
NEW TASK GENERAL INFO
415
NEW TASK DEPENDENCY INFO
420
NEW TASK RESOURCE INFO
425
NEW TASK TIME FRAME
430
CANDIDATE SCHEDULE GENERATION
435
CANDIDATE SCHEDULE
440
CANDIDATE SCHEDULE PERFORMANCE SCORE
445
COMPARISON TO EXISTING SCHEDULE PERFORMANCE SCORE

FIG. 4

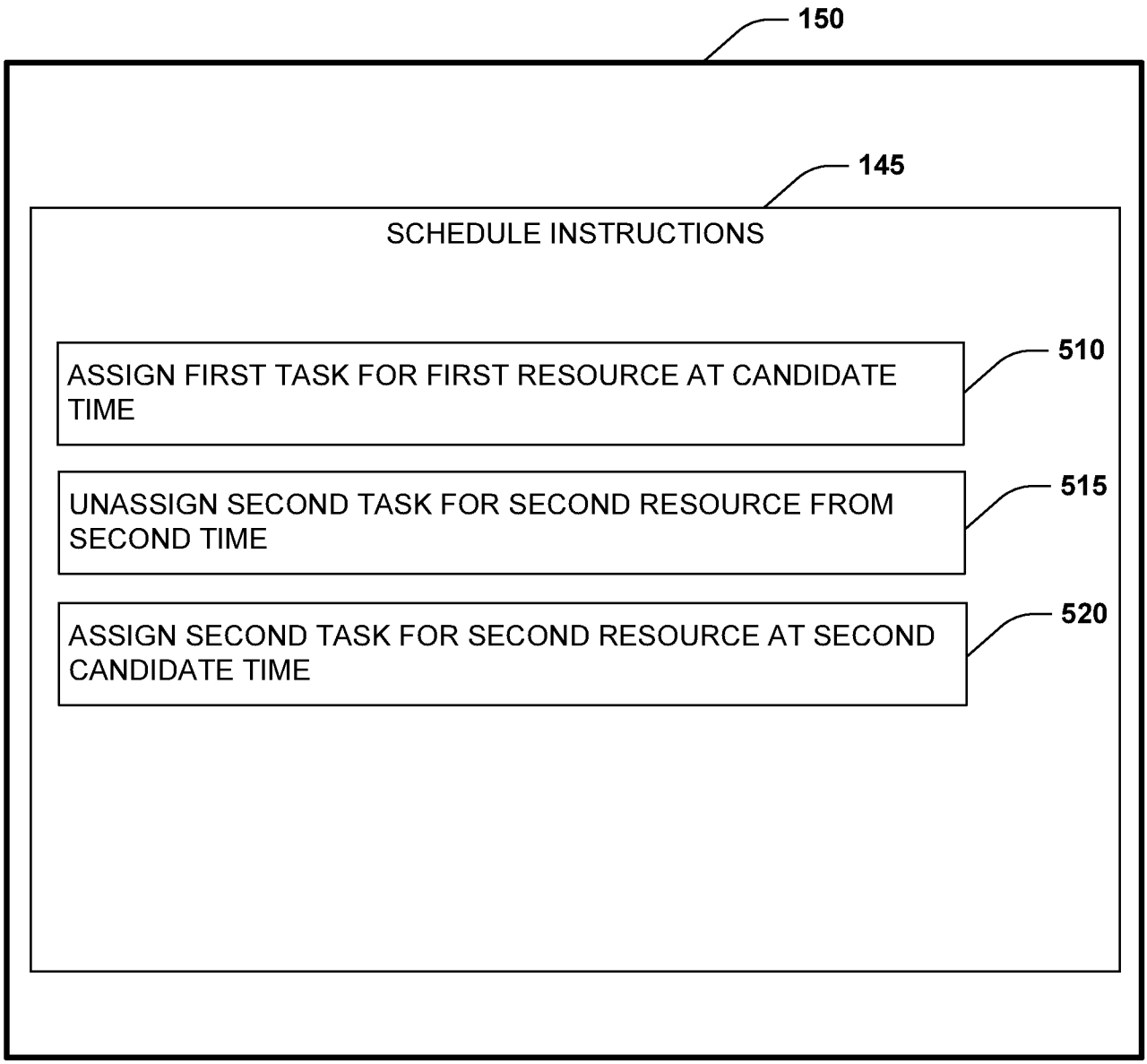
150
145
SCHEDULE INSTRUCTIONS
ASSIGN FIRST TASK FOR FIRST RESOURCE AT CANDIDATE TIME
510
UNASSIGN SECOND TASK FOR SECOND RESOURCE FROM SECOND TIME
515
ASSIGN SECOND TASK FOR SECOND RESOURCE AT SECOND CANDIDATE TIME
520

FIG. 5

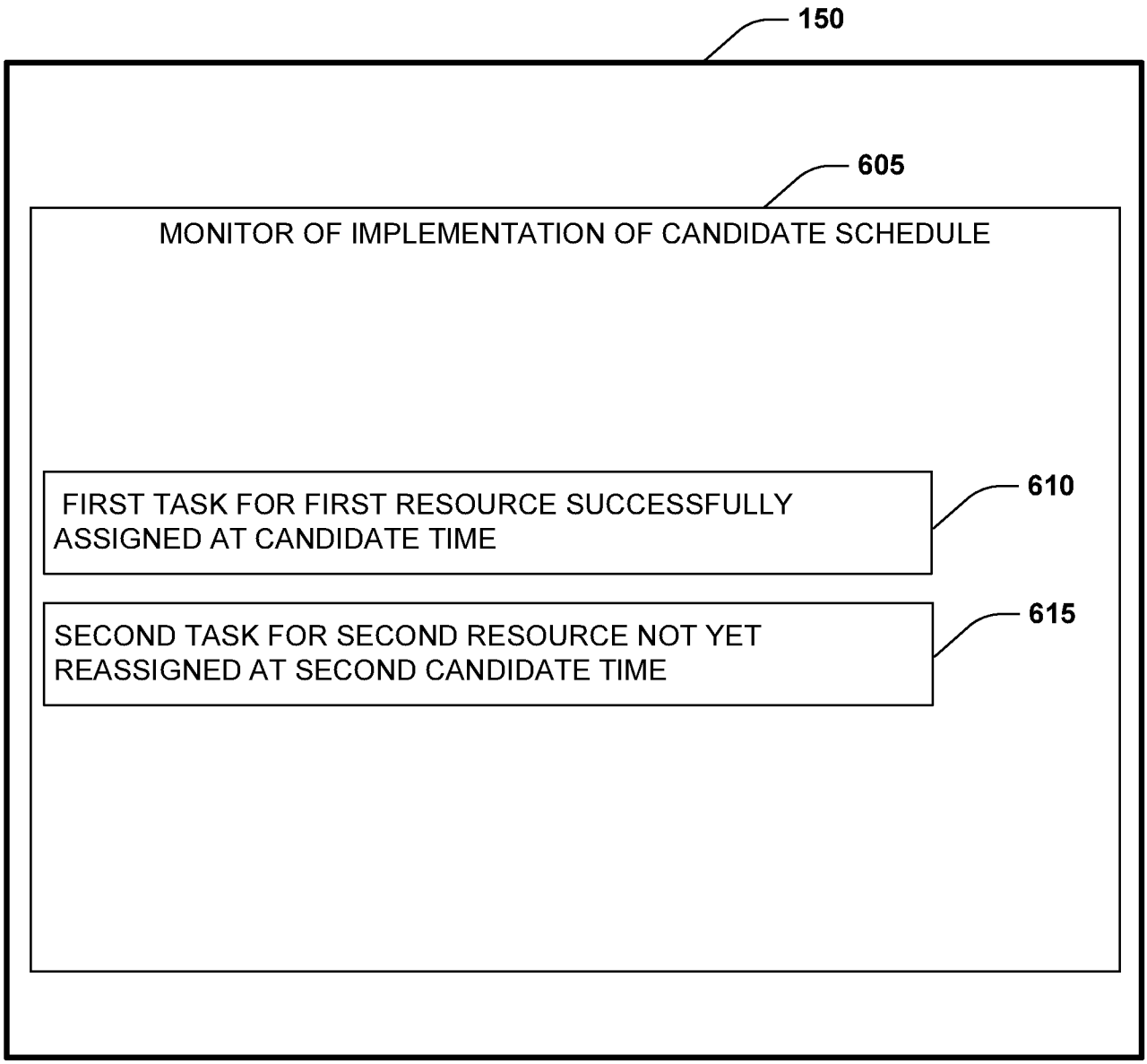
150
605
MONITOR OF IMPLEMENTATION OF CANDIDATE SCHEDULE
FIRST TASK FOR FIRST RESOURCE SUCCESSFULLY ASSIGNED AT CANDIDATE TIME
610
SECOND TASK FOR SECOND RESOURCE NOT YET REASSIGNED AT SECOND CANDIDATE TIME
615

FIG. 6

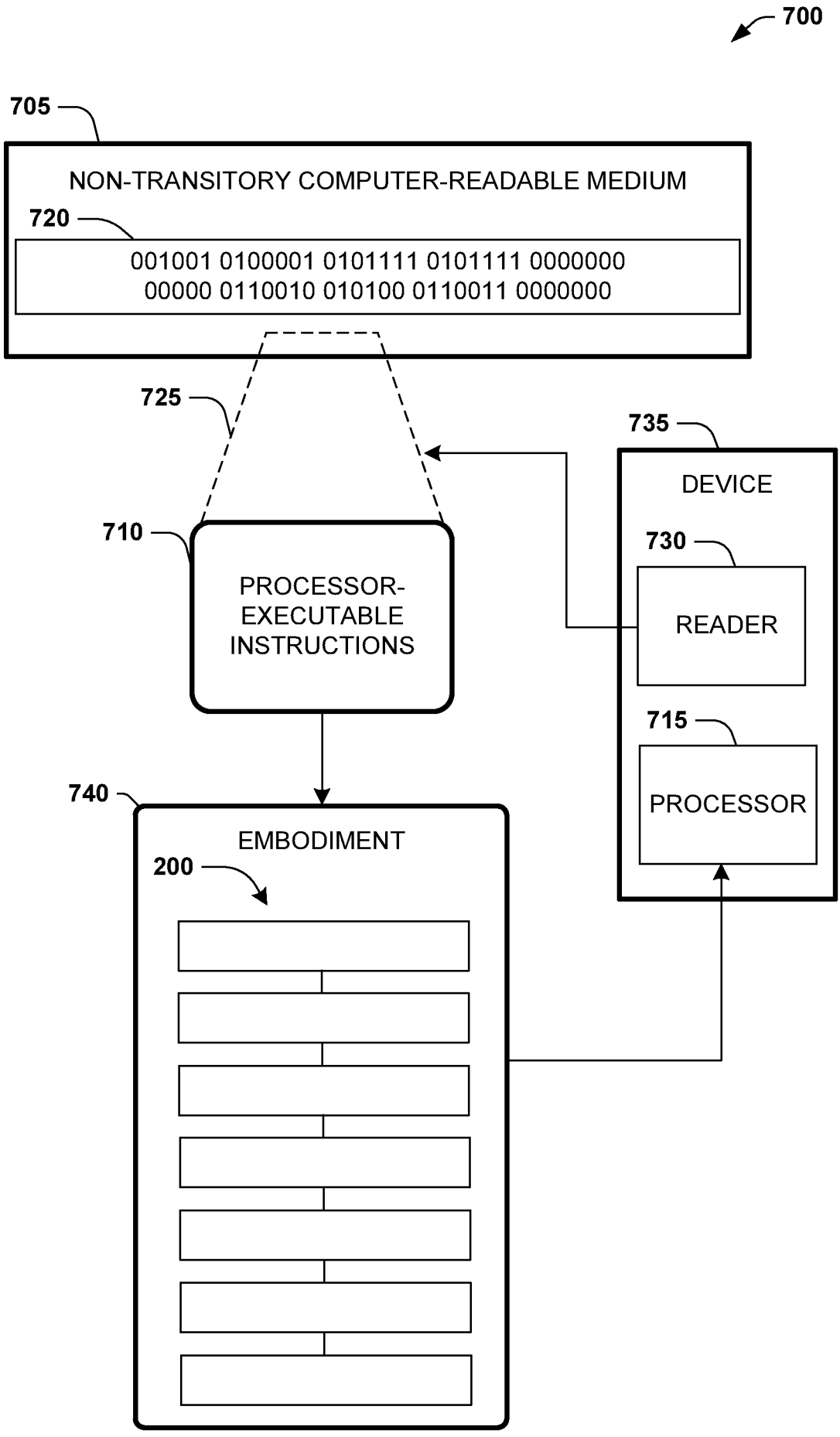
700
705
NON-TRANSITORY COMPUTER-READABLE MEDIUM
720
001001 0100001 0101111 0101111 0000000
00000 0110010 010100 0110011 0000000
725
735
DEVICE
710
PROCESSOR-
EXECUTABLE
INSTRUCTIONS
730
READER
715
PROCESSOR
740
EMBODIMENT
200

FIG. 7

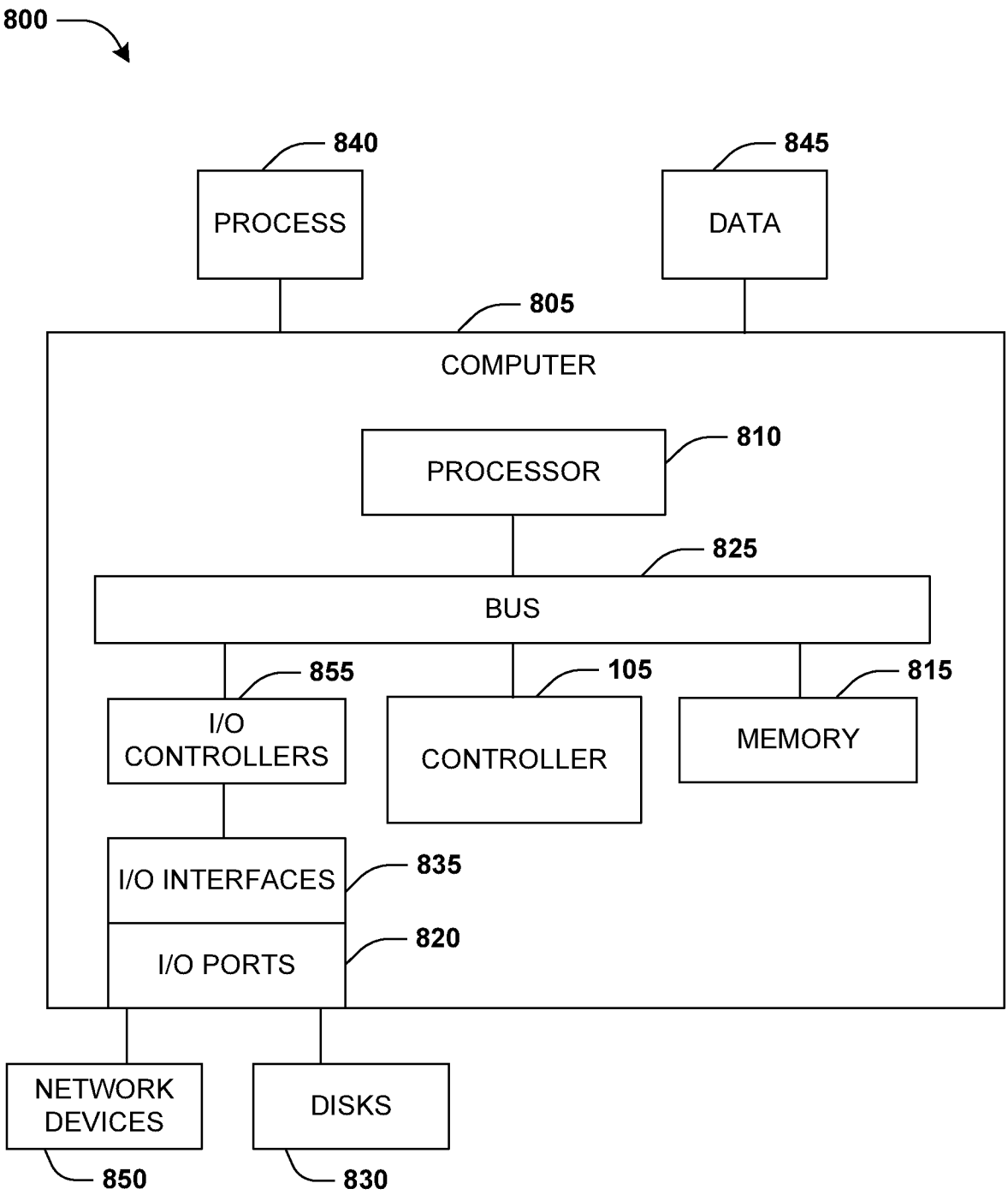
800
840
PROCESS
845
DATA
805
COMPUTER
PROCESSOR
810
825
BUS
855
I/O CONTROLLERS
105
CONTROLLER
815
MEMORY
I/O INTERFACES
835
I/O PORTS
820
NETWORK DEVICES
850
DISKS
830

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/034338

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06Q10/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2016/026935 A1 (BOTEA ADI I [IE] ET AL) 28 January 2016 (2016-01-28) figures 1-5 paragraphs [0003] - [0005] paragraphs [0018] - [0044] paragraphs [0066] - [0096] -----	1-15
X	US 2016/307145 A1 (BANERJEE DIPYAMAN [IN] ET AL) 20 October 2016 (2016-10-20) figures 4-6 paragraphs [0014] - [0023] paragraphs [0024] - [0048] paragraphs [0059] - [0096] ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 July 2018	03/08/2018

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Mülthaler, Evelyn

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/034338

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 8 543 438 B1 (FLEISS JOEL E [US]) 24 September 2013 (2013-09-24) figures 1A, 3A-D, 6A column 3, line 51 - column 7, line 17 column 12, line 49 - column 20, line 11 -----	1-15
X	US 2011/161964 A1 (PIAZZA JEFF [US] ET AL) 30 June 2011 (2011-06-30) figures 1, 2, 6 paragraphs [0021] - [0029] paragraphs [0045] - [0053] -----	1-15

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2018/034338

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2016026935 A1	28-01-2016	NONE	
US 2016307145 A1	20-10-2016	NONE	
US 8543438 B1	24-09-2013	NONE	
US 2011161964 A1	30-06-2011	US 2011161964 A1	30-06-2011
		US 2015033237 A1	29-01-2015